

LASAL OS

Betriebssystem

Betriebsanleitung

Herausgeber: SIGMATEK GmbH & Co KG

A-5112 Lamprechtshausen

Tel.: +43/6274/4321

Fax: +43/6274/4321-18

Email: office@sigmatek.at

WWW.SIGMATEK-AUTOMATION.COM

Copyright © 2004
SIGMATEK GmbH & Co KG

Originalsprache

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder in einem anderen Verfahren) ohne ausdrückliche Genehmigung reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Inhaltliche Änderungen behalten wir uns ohne Ankündigung vor. Die SIGMATEK GmbH & Co KG haftet nicht für technische oder drucktechnische Fehler in diesem Handbuch und übernimmt keine Haftung für Schäden, die auf die Nutzung dieses Handbuchs zurückzuführen sind.

Table of Contents

LASAL OS	37
Benutzerhandbuch	38
Benutzerhandbuch	38
Software-Struktur einer LASAL CPU	38
Der Loader	40
Comlink-Schnittstelle	43
Aktualisierung des Betriebssystems	46
Backup von Persistenz-Daten	46
Zuweisung von Laufwerk-Zeichen	46
Protokollierung von System- und Applikationsnachrichten	50
Betriebssystem-Nachrichten	51
Runtime-Check und Watchdog	58
Hardware Watchdogs	58
Software-Tests	59
Fehlererkennung	60
Operationsablauf	62
CIPC Peripherie-Reset	63
Peripherer DIAS Slave Reset	64
Einfacher Router	65
Downgrade Sicherheit	66
Von LASALOS unterstützte USB-Geräte	67
Von LASALOS unterstützte USB-Geräte für CCL722, DCL642 und DTC081	68
USB-Update	68
Setup/Defragmentierung Compact-Flash	71
Lasal OS Tasks	72
Anlegen von CAN-Identifiers	74
SPS (Online + Comlink Protokoll)	74
MiniDisplay	75
Aufzählen von Ethernet-Schnittstellen-Verbindungen	75
Status- und Fehlermeldungen	76
Routing	84

Übersicht	84
Kommandos	84
Routing-Tabelle Syntax	84
Boot-Screen	86
Dateisystem für USB-Sticks	87
Task-Management	87
Threads	87
LASAL Tasks	88
Programming Guide	90
Der Konstruktor	90
Exception Handling	92
APIs	92
OS_SSR_SetHandler	92
OS_SSR_SkipOverDIV	94
Command Line Interface CLI	96
Übersicht	96
Dateisystembefehle	96
Betriebssystembefehle	97
LASAL data file (active.dat) Befehle	97
Kernel Error Log Befehle	97
Projektbefehle	97
Konfigurationsbefehle	98
Konfigurationsbefehle (IP)	98
Konfigurationsbefehle (CAN)	98
Konfigurationsbefehle (DIAS)	98
Konfigurationsbefehle (SERIAL)	98
Konfigurationsbefehle (Maus/Touch)	98
Konfigurationsbefehle (Drucker)	99
Batch-Verarbeitungsbefehle	99
Umgebungsvariablen	99
Command Line Syntax	99
Tasten und Dateien	99
Reihenfolge der Ausführung von Befehlen in autoexec.lsl	100

Dateisystembefehle	100
ARCH	101
ATTRIB/ATTR	102
CD/CHDIR	103
CHKDSK	104
COPY/C	105
CREATE	106
DEL/ERASE	106
DIR/D	107
DRIVES	109
FC	109
FORMAT	109
LABEL	110
MD/MKDIR	110
MOUNT	111
PARTITION	112
RD/RMDIR	112
REN	113
SMBSVR	113
TREE	115
TYPE	115
UMOUNT	115
UNZIP	116
VOL	116
XCOPY	117
ZIP	117
Betriebssystembefehle	118
ADDR	118
APPTASKS	118
ARP	118
BGTASKS	119
BIOS	119

BOOT	119
BOOTPIC	121
BOXES	122
BUFFERS/BUF/B	122
CANINFO	122
CIL	122
CYTASKS	123
DATE	123
DUMP	123
ERRORLOG	124
EXCEPT	124
FCLOSEALL	124
FILE	124
FILES	125
FLUSHLOG	126
GET ERRORLOG	126
HANDLES	126
HEAP	127
HELP/?	127
INFO	128
INTS	128
LINK	129
LISTTS	130
LOG	131
MEM	131
MEMDUMP	131
NETSTAT	132
NETWORK	132
NUMLOCK	132
PACKAGE	133
PING	134
PNPBIOS	134

QUCMD	134
REBOOT	135
REXX	135
ROUTE	135
RTTASKS	136
SEMAS	136
SET CYCLIC	136
SET DBGHEAP	137
SET DEBUG	139
SET ERRORLOG	139
SET EVENTLOG	140
SET SYSTRACE	141
SET TRACEBUFSIZE	143
SET USB_CACHE_FLUSHING	143
SETENV	144
SETLEDSTATUS	145
SHOWENV	145
SRAMLOAD	145
SRAMSAVE	146
STATUS	146
TASKS1/TASKS	146
TASKS2	146
UNMOUNT	147
VER	147
vpndaemon	147
VT	149
Projektbefehle	151
SET FPUROUNDCONTROL	151
LSLLOAD/LO	152
LSLSAVE/SA	152
RUN	152
RESET	153

PROJECT	153
Konfigurationsbefehle	154
ADDBR	154
CALIB	155
Calib check	155
CHANGECERT	155
DIAS	156
EURO	157
FIREWALL	157
IP	160
IPCINFO	161
SCREENSAVER	161
SET	162
SET APPCODE	162
SET APPDATA	163
SET APPLSAVE	163
SET BOOTTEXT	164
SET BTRUNTIME	164
SET CAN	164
SET CLI	165
SET CPUBT	166
SET CPUTC	166
SET CPURT	166
SET DATE	167
SET DBGLEVEL	167
SET DIASERROR	172
SET DISPLAY	172
SET DISPLAYRESOLUTION	173
SET DISPLAYROTATE XX	174
SET FIRSTUSBDRIVE	174
SET FORCEXTSINGLE	175
SET IP	175

SET IP PORT	177
SET KEYS	177
SET LASALCOM	178
SET LCD	179
SET MAINTIMER	179
SET MOUSE	180
SET MULTICOREOJBS	181
SET MULTI_VM	182
SET OSHEAP	183
SET OSZIMEM	183
SET PRINTER	184
SET RAM	184
SET ROUTINGTABLE	185
SET RTRUNTIME	185
SET RUNTIME	185
SET SECURE	186
SET SRAMRETAIN	186
SET STATION	186
SET STOPWAIT_TIME	187
SET TIME	187
SET VISU	188
SET WATCHDOG	188
TOUCHCFG	188
TOUCHTEST	188
UPDATETOUCH	189
US	189
Batch-Verarbeitungsbefehle	189
CLS	189
DELAY	189
ECHO/@ECHO	190
EXIT	190
GOTO	190

IF EXISTS	191
PAUSE	191
REM	191
Umgebungsvariablen	192
FILENAMES	192
REBOOTONPF	192
LasalOS Dateifehler Rückgabe-Codes	192
Software-Konfiguration	196
Software-Komponenten	196
Installation der Debug-Tools LASAL Class und LSE	196
Installation über das Command Line Interface (CLI)	197
Automatische Installation von austauschbaren Speichermedien	198
Rexx-Programme zur Installation von Software	199
Beschreibungsdatei eines Software-Pakets	199
Beispiele für Beschreibungsdateien	201
Programm zur Erstellung einer Archivdatei	202
AUTOSTRT REXX-Programm für die Installation des Software-Packages	203
Beispiel	203
System Update IPC	203
System Update CPU	204
LASAL auf mehreren Datenträgern	205
Rexx Interpreter	208
Was ist LASALOS Rexx?	208
Der Zweck dieses Dokuments	208
Integrierte LASALOS-spezifische Funktionen	211
DIRLIST	212
DRIVEINFO	213
FILEINFO	214
GETCH	215
GETCHE	215
GETCPUSTATUS	216
GETDATAD	216

GETOSVERSION	216
GETOSVERSION2	217
GETRES	218
HWINFO	219
HWINFOONLINE	219
KBHIT	220
LSLLINK	220
LSLLOADPRJ	221
LSLSEND	221
LSLSENDMOD	222
LSLSENDMODCLSNAMEx	222
MILLISLEEP	223
OFFLINE	223
ONLINE	223
RESETSPS	224
REXEC	224
RUNSPS	225
RXFILE	225
SENDOS	226
SERVERREAD	226
SERVERWRITE	227
SETBREAKPOINT	227
TXFILE	228
Command Line Interface (CLI) Befehle	229
Anhang A - Beispiel Rexx-Programme	229
Anhang B - Fehler-Codes	232
Anhang C - REXX How To	233
Funktionsaufruf	233
Bedingungen und Schleifen	235
Online-Funktionen	236
ONLINE(interface, address)	237
OFFLINE(online-descriptor)	237

Beispielfunktion	237
CLI-Kommandos ausführen	239
Address System cmd	239
REXECINTERFACE, ADDRESS, CMD)	239
CLI-Kommando: SHOWENV	240
Pipe Operator „>“	240
GETDATAD(interface, address, mem-addr, length, stem)	241
Dateisystemoperationen	242
stream(stream, option, command)	242
lineout(stream, string, line)	243
linein(stream, line, count)	244
lines(stream, option)	244
attrib(path, option)	245
REXX Fehler-Codes	246
-307 – Negatives Kommando empfangen	246
Nützliche Code-Abschnitte	246
Konvertieren	246
Lesen und Verarbeiten	247
Logdatei erstellen	248
Erstellen eines Pfads	251
Kopieren eines Ordners 1	253
Kopieren eines Ordners 2	256
Kopieren eines Ordners 3	258
Notepad++ Highlighting	264
Links	267
Code Highlighting	267
VNCcli	275

Was ist LasalOS VNCcli?	275
Der Zweck dieses Dokuments	275
LASAL OS VNCcli-API Funktionen	276
Die OS-Schnittstelle VNCCLI	276
Callback-Funktion	277
udSize	280
udVersion	280
udVersionVNC	281
VNCHandle	281
VNCCreate	282
VNCDestroy	284
VNCDestroyHandle	284
VNCDraw	285
VNCDraw2	286
VNCExit	287
VNCForceRepaint	287
VNCFreeBuffer	288
VNCFreeOldSockets	288
VNCGetSize	289
VNCReset	289
VNCSendHIDEEvent	290
VNCSendKbrdEvent	291
VNCSendKeyboardEvents	291
VNCSendPointerEvents	292
VNCSetCliSleep	293
VNCSetConnectTimeout	294
VNCSetKaValues	294
VNCSetOptions	295
VNCSetTaskDelayTime	297
Fehlermeldungen in der Logdatei Event00.log	298
Fehlermeldungen VNC Client	300
Command Line Interface (CLI) Befehle	301
VNCsvr	301

Was ist LasalOS VNCSvr?	301
Der Zweck dieses Dokuments	302
LASAL OS VNCSvr-API Funktionen	303
Die OS-Schnittstelle VNCSVR	303
udSize	304
udVersion	304
udVersionVNC	305
VNCSVRAcceptKeyboardEvents	305
VNCSVRAcceptPointerEvents	306
VNCSVRAlwaysShared	306
VNCSVRClientWaitTimeMillis	307
VNCSVRCompareFB	308
VNCSVRDisconnectAllClients	309
VNCSVRDisconnectClient	309
VNCSVRConnectedClients	309
VNCSVRConnectedClients2	310
VNCSVRDisconnectClientsOnNonsharedConnection	310
VNCSVRDisableSharedMemory	311
VNCSVREnumConnections	312
VNCSVREnumConnections2	313
VNCSVRExit	314
VNCSVRFullRefreshCycle	314
VNCSVRInit	315
VNCSVRInitialized	316
VNCSVRMaxConnections	316
VNCSVRNeverShared	317
VNCSVRReset	317
VNCSVRSetDisplayOffset	318
VNCSVRSetFilter	318
VNCSVRSetPass	319
VNCSVRTimeoutInactiveClient	320
VNCSVRUpdateRect	321

Fehler-Codes VNC Server	321
Befehle der Befehlszeilen-Schnittstelle	322
Konfigurieren des VNC-Servers mit Umgebungsvariablen	322
VNC_SVR_ACCEPT_KEYBOARD	323
VNC_SVR_ACCEPT_POINTER	323
VNC_SVR_ALWAYS_SHARED	323
VNC_SVR_BLACKLIST_LEVEL	324
VNC_SVR_COMPARE_FB	324
VNC_SVR_DESKTOP_NAME	324
VNC_SVR_DISCONNECT_ON_NONSHARED	325
VNC_SVR_DISABLE_SHARED_MEMORY	325
VNC_SVR_FILTER	325
VNC_SVR_FORCE_BPP	326
VNC_SVR_FULL_UPDATE	326
VNC_SVR_LOGFILE	326
VNC_SVR_LOGSTR	327
VNC_SVR_MAX_CONNECTIONS	327
VNC_SVR_NEVER_SHARED	327
VNC_SVR_PORT	328
VNC_SVR_PORT2	328
VNC_SVR_TIMEOUT_INACTIVE	328
VNC_SVR_WAIT_FOR_CLIENT	328
Repeater-Funktion	329
Starten der Repeater-Funktion	329
Verwenden der Repeater-Funktion	330
Konfigurieren der Repeater-Funktion	331
Beispielkonfigurationen mittels AUTOEXEC.LSL	331
Szenario 1: Der Server sollte genau eine Verbindung von einem bestimmten Host zulassen	331
Szenario 2: Der Server soll eine Verbindung mit jedem Host erlauben, aber nur zur Anzeige	332
Szenario 3: Einrichten des Servers unter Verwendung einer separaten Protokolldatei	332
Szenario 4: Einen Server einstellen, der Remote-Events in das Benutzerprotokoll event02.log schreibt	333

Beispielkonfiguration mit der Datei vncsvr.cfg	333
Szenario 1: VNC-Server meldet sich mit der Seriennummer an	333
Szenario 2: VNC-Server meldet sich mit einer individuellen ID an	333
Szenario 3: VNC-Server meldet sich mit DNS an	333
SRAM	334
Formate	334
Format 1	334
Format 2	334
Konvertieren zwischen Formaten	335
Remanente Server	335
RamEx	335
Speichern, wiederherstellen, löschen	336
SRAM sichern	336
SRAM löschen	336
Laden einer Reorganisationskopie	337
LASAL CLASS Tool Ram Image	337
SRAMDisk	337
SRAMDisk Backup-Mechanismen	339
Sichern von Daten während des Betriebs und beim Ausschalten	339
Sichern von Daten nur beim Ausschalten	340
Vorteile und Nachteile dieser beiden Arten der Datensicherung	340
Überwachung der Anzahl von Schreibvorgängen auf der SD-Karte	341
Erkennen eines Fehlers beim Schreiben auf die SRAMDisk während des Herunterfahrens	341
Reaktion, wenn ein Fehler im SRAM erkannt wird	342
SRAM reorganisieren	342
Größe und Speicheraufteilung des SRAMs	343
Repeater	344
LASAL OS mit Repeater-Funktionalität	344
Konfiguration	344
Islrepeater.cfg	345
Kommando	346

Beispiel-Konfigurationen mit der lslrepeater.cfg-Datei	346
Szenario 1: Steuerung meldet sich mit Seriennummer an	346
Szenario 2: Steuerung meldet sich mit individueller ID an	346
Szenario 3: Steuerung meldet sich über DNS an	347
LASAL Tools	347
LASAL Repeater	348
 Remote Diagnose	348
Betriebssystem-Schnittstellen	352
Variablen	352
Inhalt der OP-Struktur	355
Inhalt der _OnlineMap-Struktur	357
API MultiTask	357
Allgemeine Hinweise	357
Multitasking	357
Zeit	364
Message Passing	365
Semaphore Handhabung	366
Mailbox Handhabung	368
Multitask-Klasse: Interface-Funktionen	368
Funktionen	368
Erweiterte Funktionen	369
GETLASTERROR	369
GETTIME	370
Task-Funktionen	371
CREATETHREAD	371
CURRENTTASKHANDLE	373
DELAYUNTIL	374
GETMINSTACK	374
GETTASKPRIORITY	375
GETTASKSTACK	375

GETTASKSTATE	376
RECEIVE	377
RECEIVECOND	378
RECEIVETIMED	378
RESUME	379
SEND	379
SENDCOND	380
SENDTIMED	381
SETPRIORITY	381
SUSPEND	382
TASKDELAY	382
TERMINATETASK	383
 Semaphore-Funktionen	384
CREATESEMAPHORE	384
DELETESEMAPHORE	385
PULSE	386
RESETEVENT	386
RESOURCEOWNER	387
SEMAVALUE	387
SIGNAL	388
WAIT	389
WAITCOND	390
WAITTIMED	390
 Mailbox-Funktionen	391
CLEARMAILBOX	391
CREATEMAILBOX	392
DELETEMAILBOX	392
GET	393
GETCOND	393

GETTIMED	394
MESSAGES	395
NEXTCOND	395
PUT	396
PUTCOND	396
PUTFRONT	397
PUTFRONTCOND	397
PUTFRONTTIMED	398
PUTTIMED	399
Schnittstellen-Funktionen	400
Beispiele	401
API Application Heap	404
Übersicht	404
Debug-Heap verwenden	404
SSR-Interface Funktionen für den Heap	405
OS_SSR_Malloc	405
OS_SSR_ReAlloc	406
OS_SSR_Free	407
API Serielle Schnittstelle	407
Übersicht	407
Interface-Funktionen SERIAL	408
SERUSER_ClearRecvBuffer	408
SERUSER_Close	408
SERUSER_EnableFIFO	409
SERUSER_Get422Mode	410
SERUSER_GetError	411
SERUSER_GetInfo	412
SERUSER_GetModemControl	414
SERUSER_GetModemStatus	414
SERUSER_GetRecvStatus	415
SERUSER_GetSendStatus	416
SERUSER_Init	416
SERUSER_RecvBlock	418

SERUSER_RecvChar	419
SERUSER_Send	420
SERUSER_Set422Mode	420
SERUSER_SetBufferRecv	421
SERUSER_SetModemControl	422
SERUSER_SetOnline	423
SERUSER_UserFunction	424
API Seriennummer	425
Allgemein	425
Strukturen, die in ISYSSERNUM verwendet werden	425
tagPLCInfo	425
Interface-Funktionen ISYSSERNUM	427
SernumGetPLC	427
SernumGetPLCDrive	428
SernumGetPLCInfo	429
API Sysmsg-Schnittstelle	430
Übersicht	430
Interface-Funktionen SYSMSG	431
OS_SYSMSG_LCLOSE	431
OS_SYSMSG_LCREATE	432
OS_SYSMSG_LFLUSH	433
OS_SYSMSG_LINFO	434
OS_SYSMSG_LOPEN	434
OS_SYSMSG_LPRINTFLN1-4	435
OS_SYSMSG_LWRITE	436
OS_SYSMSG_WRITE_I	437
OS_SYSMSG_LWRITELN	438
Sysmsg – Änderungen im neuen Betriebssystem	438
Erweiterte Userlog Funktionen	439
OS_USERLOG_EXPORT	440
OS_USERLOG_FILEHEADER	440
OS_USERLOG_FLUSH	441
OS_USERLOG_LAST_EXPORT_RESULT	442

OS_USERLOG_PRINTFNL (1-2)	443
OS_USERLOG_WRITEEVENT	444
API TCP Benutzer-Schnittstelle	445
Fehlerwerte	445
Anwendung der OS-Funktionen	448
OS TCP USER-FUNKTIONEN	449
OS_TCP_USER_ACCEPT	449
OS_TCP_USER_CLOSESOCKET	451
OS_TCP_USER_CONNECT	452
OS_TCP_USER_GETCOMLINKPORT	453
OS_TCP_USER_GETERRORSTRING	453
OS_TCP_USER_GETLINKSTATUS	454
OS_TCP_USER_GETPEERIP	456
OS_TCP_USER_GETPEERPORT	457
OS_TCP_USER_GETSERVBYNAME	457
OS_TCP_USER_GETSOCKIP	459
OS_TCP_USER_GETSOCKPORT	460
OS_TCP_USER_IOCTLSOCKET	461
OS_TCP_USER_IPINFO	462
OS_TCP_USER_LISTEN	464
OS_TCP_USER_NREAD_AVAILABLE	465
OS_TCP_USER_PING	466
OS_TCP_USER_RECV	467
OS_TCP_USER_RECVFROM	468
OS_TCP_USER_SELECT	470
OS_TCP_USER_SEND	472
OS_TCP_USER_SENDTO	473
OS_TCP_USER_SETSOCKOPT	475
OS_TCP_USER_SHUTDOWN	482
OS_TCP_USER_SOCKET	483
OS_TCP_USER_SOCKET_EX	484
OS_TCP_USER_STRTOULONG	485
OS_TCP_USER_TOIP	486

OS_TCP_USER_ULONGTOSTR	487
OS_TCP_USER_STRTOULONG_ASY	488
OS_UDP_USER_BIND	489
OS_UDP_USER_SOCKET	490
Defines	491
#define IP_ADDR_ANY	491
#define IP_ADDR_BROADCAST	491
#define IP_OPT_ADDR	491
#define IP_OPT_ETHERNET_ADDR	491
#define IP_OPT_SUBNETMASK	491
Aufzählung von Ethernet-Schnittstellen	492
Beispiele	492
API XTimer	495
Allgemein	495
Interface-Funktionen XTIMER	495
XtimerInit	495
XTimerSetInterval	497
XTimerSetMode	498
XTimerStopAllUser	499
XTimerValue	500
XtimerReset	501
XtimerStop	502
XTimerStart	502
Web Server	503
Befehl	503
Anforderungen	503
Installation	503
Sicherheit	504
IP-Adresse Authentifizierung	504
Authentifizierung mit Benutzername und Passwort	505
Zusätzliche Funktionen	506
Command Line Interface (CLI) Befehle	506
WEBSVR, WEBSVR INFO	506

WEBSVR START	507
WEBSVR AUTH	507
WEBSVR FILTER	511
WEBSVR STOP	513
Umgebungsvariablen	513
WEBROOT	513
WEBMAXAUTH	513
WEBDEFPWD	514
Beispielkonfiguration	514
LASAL OS Web Server API	515
OS_WEB_FindStringInBuffer	516
OS_WEB_GetErrorStringByValue	517
OS_WEB_GetLineFromBrowser	517
OS_WEB_InetStrToByte	519
OS_WEB_RegisterGetCallback	520
OS_WEB_RegisterPostCallback	522
OS_WEB_SendLineToBrowser	525
OS_WEB_SetAuthentication	526
OS_WEB_SetBrowserList	528
Quick Review	530
Fehler-Codes	530
API-Beispiel	531
HTML Quelldatei: index.htm	531
HTML Quelldatei: ApplCGI.htm	531
HTML Quelldatei: Err401.htm	533
LASAL Class Projekt WebServer	533
LASAL Class Projekt WebServer Structure Text source code	534
Einstellungen Autoexec.lsl	539
Fehler-, Warnung- und Info-Protokollierung	540
Anhang	540
Typen	540
Fehlerwerte	541
Flags	541

OS Interface Library Klassen	543
Memory-Library Klassen	543
RamFile	543
Schnittstellen	543
Globale Methoden	544
GetDataAt	544
GetDataAtBackground	544
GetFileState	545
GetUserVersion	545
SetDataAtBackground	545
SetDataAt	546
SetSize	546
SetSizeBackground	547
SetUserVersion	547
Checksum Berechnung	547
Encoding	547
RamFileRingBuffer	548
Schnittstellen	548
Globale Methoden	549
GetDataRB	549
GetDataRBBackground	549
GetLastDataRB	550
GetLastDataRBBackground	550
GetNumberOfValidEntries	551
RecordDataRB	551
RecordDataRBBackground	551
RecordRingbufferSize	552

RecordRingbufferSizeBackground	552
RecordUserVersion	553
Checksum Berechnung	553
Encoding	553
LARS	554
LARS	554
Einleitung	554
Systemanforderungen	554
Installation	554
Konfiguration	554
LARSConfigTool	555
Settings Wizard	556
Pfad-Einstellung	557
Projekt-Einstellung	559
Window-Einstellung	561
Experteneinstellungen	562
Reset Window Position	562
Autoexec.lsl bearbeiten	562
Screen-Projekt Server-Stationen pingen	562
LARS Memory and Port Setting	563
LARS Drive Mapping	564
Repeater Connection Configuration	565
Verwendung von LARS	566
Kommunikationsschnittstellen	566
Programmierhinweise	567
Drucken mit LARS	568
Kommunikation und Treiber	569
Online-Kommunikation	569
DebugIP-Kommunikation	569
Comlink	570
Online-Kommunikation	573
Kommunikationstreiber Lasal32 API	573

Übersicht	573
Lasal32 API für Windows	573
Lasal32 API für Linux	573
Lasal32 API-Funktionen	573
Fehler-Behandlung	574
Verwaltung der Online-Verbindung	576
IsOnline	576
IsOnlineH	576
LslPing	577
OcbClose	578
OcbOpen	578
Offline	578
OfflineH	578
Online	579
OnlineH	582
OnlineOptions	583
OnlineOptionsH	583
OnlinePwd	584
OnlinePwdH	585
OnlineSSL	585
OnlineSSLH	585
Modemfunktionen	587
ModemOpen	588
ModemGetNumber	588
ModemClose	588
ModemCall	591
ModemGetName	591
Funktionen zum Datenaustausch	592
GetData	593
GetDataH	594
GetDataList	594
GetDataListH	595
LslExecKillH	596

LslExecNewInstrEx	598
LslExecNewInstrExH	598
LslGetAdressVar	600
LslGetAdressVarH	600
LslGetDataEx	601
LslGetDataExH	601
LslGetDataListEx	602
LslGetDataListExH	602
LslGetDataListSpecial	604
LslGetDataListSpecialH	604
LslSetDataEx	605
LslSetDataExH	605
SendDataList	606
SendDataListH	607
SetData	608
SetDataH	609
Status-Informationen und SPS-Befehle	609
GetBusType	610
GetBusTypeH	610
GetChk	611
GetChkByType	611
GetChkByTypeH	611
GetChkH	612
GetCpuStatus	613
GetCpuStatus2	613
GetCpuStatus2H	613
GetCpuStatusH	614
GetPowerFailInfo	615
LslGetDllInfo	616
LslGetDllVersion	617
LslGetMemInfo	617
LslGetMemInfoH	617

LslGetPLCInfo	620
LslGetPLCInfoH	620
LslGetProjectInfo	622
LslGetProjectInfoEx	622
LslGetProjectInfoExH	623
LslGetProjectInfoH	625
LslReadDateTime	626
LslReadDateTimeH	626
LslSetDateTime	627
LslSetDateTimeH	627
LslSyssernumCommand	627
LslSyssernumCommandH	628
SetCommand	629
SetCommandH	629
Administrativfunktionen	630
SetMultiThreadSupport	630
Dateifunktionen	631
CB_FORMAT_FUNCTYPE	631
FileInfo	632
FileInfoH	633
FileLoad	634
FileLoadH	634
FileSave	636
FileSaveEx	637
FileSaveExH	637
FileSaveH	640
FileSetAttributes	643
FileSetAttributesH	643
LslCheckDisk	644
LslCheckDiskH	645
LslFileDelete	646
LslFileDeleteH	647

LslFindCloseH	648
LslFindFirstH	648
LslFindFirst, LslFindNext, LslFindClose	650
LslFindNextH	650
LslFileTransfer	651
LslFileTransferFlush	652
LslFileTransferFlushH	652
LslFileTransferH	652
LslFileTransferFromPlcBuf	655
LslFileTransferFromPlcBufH	656
LslFileTransferMakeDir	658
LslFileTransferMakeDirH	658
LslFileTransferToPlcBuf	659
LslFileTransferToPlcBufH	659
LslFormatDrive	660
LslFormatDriveH	661
LslGetDiskSpace	662
LslGetDiskSpaceH	662
LslGetDriveListShort	664
LslGetDriveListShortH	664
LslRenameFileDir	667
LslRenameFileDirH	668
Parameter einer CB_FUNCTYPE Callback-Funktion	669
Parameter einer CB_FUNCTYPE2 Callback-Funktion	669
Verwalten der RefreshListe	670
LslRefreshListAdd	680
LslRefreshListClear	681
LslRefreshListCreate	681
LslRefreshListCreateExt	682
LslRefreshListDestroy	683
LslRefreshListGetData	684
LslRefreshListGetDataSize	686

LslRefreshListGetLoaderVersion	687
LslRefreshListGetVarInfo	688
LslRefreshListIsOnline	689
LslRefreshListOffline	689
LslRefreshListOnline	690
LslRefreshListSetCallback	690
LslRefreshListSetData	692
LslRefreshListSymbolTableInit	695
LslRefreshListStartCount	695
LslRefreshListStart	696
LslSetDbgLevelEx	696
Interne Funktionen	697
LslDownloadOS	697
LslDownloadOSH	697
LslDirect	698
LslDirectCreateDOB	698
LslDirectDestroyDOB	698
LslDirectIsOnline	699
LslDirectOffline	699
LslDirectOnline	699
LslDirectReceive	700
LslDirectReceiveCount	701
LslDirectSend	701
LslDirectSetParam	702
Anhang A	702
LslExecNewInstr	711
LslExecNewInstrH	714
LslGetObjCls	715
LslGetObjClsH	715
LslGetObject	717
LslGetObjectEx	717
LslGetObjectExH	718

LslGetObjectH	719
LslGetObjectID	720
LslGetObjectIDH	720
LslReadFromClt	721
LslReadFromCltH	721
LslReadFromSvr	722
LslReadFromSvrH	722
LslReadFromSvrStr	724
LslReadFromSvrStrH	724
LslWriteToClt	725
LslWriteToCltH	725
LslWriteToSvr	727
LslWriteToSvrEx	727
LslWriteToSvrExH	727
LslWriteToSvrH	728
LslWriteToSvrStr	730
LslWriteToSvrStrH	730
Comlink API im Loader	731
Einleitung	731
Die Comlink-Schnittstelle in LASAL Class	732
Anlegen von CAN-Identifikators	733
InstallCallBack	734
LDR_SetCanWait	735
LDR_SetCanWaitRemote	736
LDR_SetRs232ComlinkParams	736
Login	738
StartStopRefresh	740
TxCommand	740
TxCommandEx	741
TxUpd	742
Fehler-Codes	743
Library für SIGMATEK Hardware-Zugriff	744
Allgemein	744

Installation	744
Anforderungen	744
CAN APIs	745
StkCanAddObject	745
StkCanClose	746
StkCanOpen	747
StkCanRxObjectAny	747
StkCanRxObject	748
StkCanSetup	749
StkCanSwReset	750
StkCanTxObject	751
DIAS APIs	752
StkDiasCheckError	752
StkDiasClose	752
StkDiasGetModID	753
StkDiasOpen	753
StkDiasReadByte	754
StkDiasReadCtrl	754
StkDiasReadWord	755
StkDiasReadRegister	755
StkDiasWriteByte	756
StkDiasWriteCtrl	756
StkDiasWriteRegister	757
StkDiasWriteWord	757
API Safety DLL	758
Allgemeine Informationen	758
Interface-Funktion ISAFETY_DLL	759
SAFETY_CHANGE_PASSWORD	760
SAFETY_CHECK_DONGLE_FW	761
SAFETY_CLEAR_CONFIG	762
SAFETY_DELETE_STATE	762
SAFETY_DOWNLOAD_FILE	763
SAFETY_DOWNLOAD_FW	763

SAFETY_FILE_GET_PRJNAME	764
SAFETY_FILE_GET_REVNBR	765
SAFETY_FILE_GET_SCPUNAME	766
SAFETY_GET_CFGSTATE	766
SAFETY_GET_IMAGE_MOD_ID_FW	767
SAFETY_GET_IMAGE_VERSION_FW	768
SAFETY_GET_MODUL_VERSION_FW	769
SAFETY_GET_RUNSTATE	769
SAFETY_GET_SAFETY_NBR	770
SAFETY_LEAVE_SERVICE_MODE	771
SAFETY_LOGIN	771
SAFETY_NEW_STATE	772
SAFETY_OPEN_CONNECTION	773
SAFETY_QUIT_ERROR	774
SAFETY_SET_CONFIGURED	775
SAFETY_SET_FILE	776
SAFETY_SET_IMAGE_FW	776
SAFETY_SET_SAFETY_NBR	777
SAFETY_SET_SERVICE_MODE	778
SAFETY_SET_TEMP_SERVICE_MODE	778
SAFETY_SET_USERPROMPT_TIME	779
SAFETY_SET_VERIFIED	779
Ablauf Projekt-Download	781
Ablauf Firmware-Download	782
Fehler-Codes	783
FTP-Server	785
Befehle	785
Schnellstart	785
Anforderungen	785
Installation	785
Sicherheit	785
Benutzernamen	786
Passwörter	786

Zugriffsberechtigung	786
Anonymer FTP	787
Server-Konfiguration	788
ANONYMOUS	788
ANONYMACCESS	788
NOFTPSINI	789
WINCOMPLISTFORMAT	789
PASVPORT	791
DEFROOTDIR	791
MAXSESSIONS	792
FORWARDSLASH	792
BACKSLASH	793
NODRIVE	793
CMDSMO	793
WRITEMO	794
READMO	794
TPRIO	795
DPRIO	795
Command Line Interface (CLI) Befehle	796
FTPSVR INFO	796
FTPSVR HELP	796
FTPSVR START	796
FTPSVR OPTION	797
FTPSVR ACCESS	798
FTPSVR USER	799
FTPSVR STOP	801
Lasal OS FTP-Server API	801
OS Interface Library Class _FTPServer	802
AddAccess	802
AddUser	804
EditAccess	805
EditUser	807
GetConfig	809
GetUpdateFTPSiniStatus	810

GetUserAccessInfo	812
GetUserExtendedInfo	813
GetUserInfo	815
RemoveAccess	816
RemoveUser	817
StartServer	818
StopServer	819
SetConfig	820
UpdateFTPSini	821
Quick Review	822
FTPS.INI	822
Fehler-, Warnung- und Info-Protokollierung	824
Anhang	824
Typen	824
Fehlerwerte	826
Flags	827
Log-Datei	828
Allgemein	828
Detaillierte Aufschlüsselung einzelner Logeinträge	828
System	828
USB	836
DIAS	837
S-DIAS	841
VARAN	849
FTP	857
VNC	858
Log19	858
CPU-Status und Fehlermeldungen	861
Fehlersuche für DIAS-Bus-Probleme	869
Fehlersuche für VARAN-Bus-Probleme	872
Error Tree	872
Hardware-Checkliste	873
SIGMATEK Device Configuration	876

Allgemein	876
Das LOGIN-Fenster	877
Bereiche in der Device Configuration	878
Titelleiste	883
Home	885
Settings	886
Network	887
CAN-Bus	891
Display	893
Config-Dateien	896
Users	898
Logout	899
Benutzerverwaltung	902
Der Anwender "Service"	903
Der Anwender "Standard"	906
Troubleshooting	907
Disclaimer	908

LASAL OS

1 Benutzerhandbuch

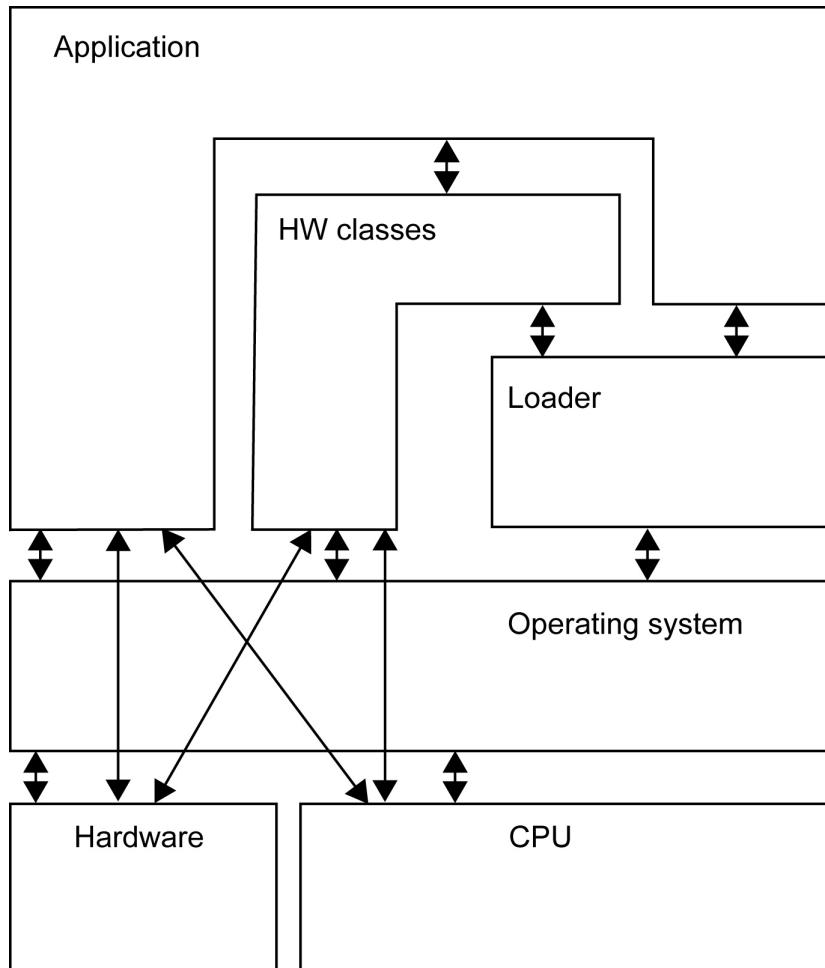
1.1 Benutzerhandbuch

1.1.1 Software-Struktur einer LASAL CPU

Eine LASAL CPU besteht aus den folgenden Software-Modulen:

- Operating system
- Loader
- Hardware-Klassen

Verwendung



(* Grafik-Software-Bereich einer LASAL CPU *)

Die Schnittstelle zwischen den einzelnen Modulen wird durch einen Pfeil gekennzeichnet.

1.1.2 Der Loader

Der Loader ist ein Bestandteil jeder Anwendung und befindet sich in jedem LASAL CLASS-Projekt. Ist LASAL CLASS installiert, wird der Loader mitinstalliert und automatisch mit dem LASAL CLASS-Projekt verbunden.

Es gibt folgende Loader-Tasks:

1) Initialisierung

Wird die Anwendung gestartet, wird die Initialisierungsfunktion des Loaders aufgerufen. Sobald die Initialisierung abgeschlossen ist, wird der zyklische Teil der Anwendung (CYwork, RTwork, Background) aufgerufen.

Die Initialisierungsphase besteht aus folgenden Abschnitten:

- Die Initialisierung des Speicherbereichs für null-spannungsgeschützte Daten.
- Die Zuordnung von Klassen und Objekten.
- Das Erstellen von Client/Server-Verbindungen.
- Die Aufteilung der Init-Werte.
- Die zyklischen Tasks in die Betriebssystem-Update-Liste eintragen.
- Aufruf von Konstruktoren.
- Aufruf von Init-Methoden.

Die Init-Methoden der einzelnen Objekte werden 12 mal aufgerufen. Bei den ersten 11 Aufrufen ist die globale Variable `_firstscan` 0. Diese wird dann beim letzten Aufruf auf 1 gesetzt.

2) Zyklischer Task

Nach der Initialisierung wird der zyklische Teil des Loaders vom zyklischen Task in 1 ms Abständen für IPCs und in 2 ms Abständen für DIAS CPUs aufgerufen.

Der zyklische Task des Loaders besteht aus folgenden Teilen:

Kommunikation mit anderen CPUs (Visu <-> CPU, Comlink)

Befehle werden zunächst gelesen, und dann beantwortet. Die Daten werden ebenfalls ohne externe Anfrage gesendet, wenn z.b. der Server-Wert in der Refresh-Liste geändert wird.

Kommunikation mit dem Programmierung-Tool

Ein Programmierungs-Tool kommuniziert mit dem Debug-Interpreter, z. B. um Server-Werte abzufragen.

Aufruf des Befehl-Interpreter-Codes (.IPR-Dateien)

Die Laufzeit des Interpreter-Codes ist auf einen maximalen Wert pro Scan (3 ms) begrenzt.

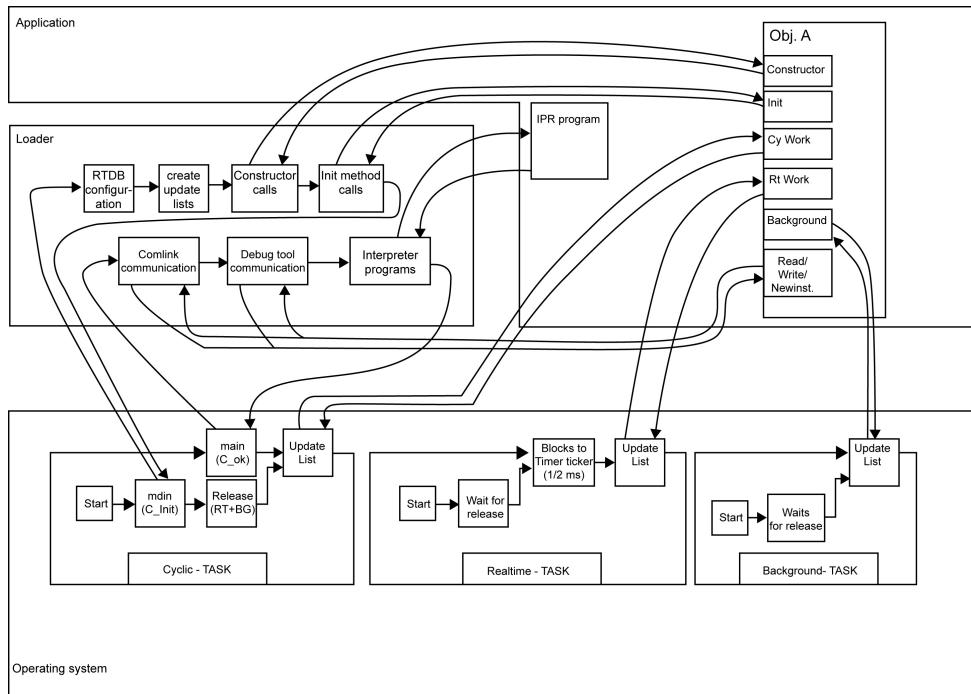
3) Funktions-Library

Es gibt einige Funktionen, die nicht für den Benutzer verfügbar sind, sondern für SIGMATEK-Klassen entwickelt wurden.

- Verwaltung der null-spannungs-geschützten Daten (RAM- und RAMEX-Zellen)
- APIs zum Comlink-Protokoll (Datenaustausch zwischen Visu und PLC)
- Zugriff auf RTDB (Datenstrukturen für Klassen und Objekte)
- Floating-Point-Funktionen

Der ST-Compiler erzeugt einen Funktionsaufruf mit Floating-Point-Operationen. Es sind Floating-Point-Funktionen verfügbar, die den Code für den mathematischen Co-Prozessor und Funktionen, die den Co-Prozessor kopieren, enthalten. Wenn Fließkomma-Funktionen ausgetauscht werden, muss das Loader-Projekt entsprechend geändert und wieder mit dem Projekt verbunden werden.

Im folgenden Diagramm wird gezeigt, wann und von welchem Task die einzelnen Objekte einer Methode aufgerufen werden.



(* Grafischer Aufruf der Objektmethoden *)

Verlauf beim Start einer Applikation bis zur Anzeige der Server-Werte

- 1) Wenn das Projekt noch nicht gespeichert ist, wird es vom Betriebssystem von der Festplatte oder Memo-Modul heruntergeladen.
- 2) Ist das Projekt noch nicht verbunden, wird es verlinkt.
- 3) Die Initialisierungsphase des Loaders wird aufgerufen.
- 4) Die zyklischen Tasks (CyWork-, RtWork- und Background-Task) werden freigegeben. Das bedeutet, dass das Betriebssystem die Update-Liste verarbeitet. Wird ein Objekt gefunden, deren zyklischer Task nicht innerhalb der eingestellten Zeit aufgerufen wurde, ruft das Betriebssystem die CyWork-, RtWork- oder Background-Methode auf. Eine Update-Liste enthält für jeden CyWork-, RtWork- und Background-Task einen Eintrag, in

dem Informationen wie ein This-Pointer, die Zykluszeit und die Uhrzeit des letzten Aufrufs gespeichert wird. Für jeden Typ eines zyklischen Tasks (Cyclic, Real-time, Background) wird eine Liste verwaltet. Der zyklische Teil des Loaders wird vor jedem Scan der zyklischen Update-Liste aufgerufen.

Der LSE-Kernel wird in der Background-Methode des _LSE-Objekts initialisiert. Die Initialisierung des Kernels besteht aus folgenden Schritten:

- LSE-Projekt laden (nur statischer Abschnitt)
- Erhalten der LASAL-ID (über Comlink-Schnittstelle)
- Aufbau der statische Refresh-Liste
- Synchronisieren und Laden der Alarme
- Anwenderschnittstelle

Während der LSE-Initialisierungsphase werden Fortschrittsbalken auf dem Bildschirm angezeigt. In diesem Zeitrahmen werden die CyWork- und RtWork-Methoden bereits ausgeführt.

6) Zyklischer Teil des LSE-Kernels

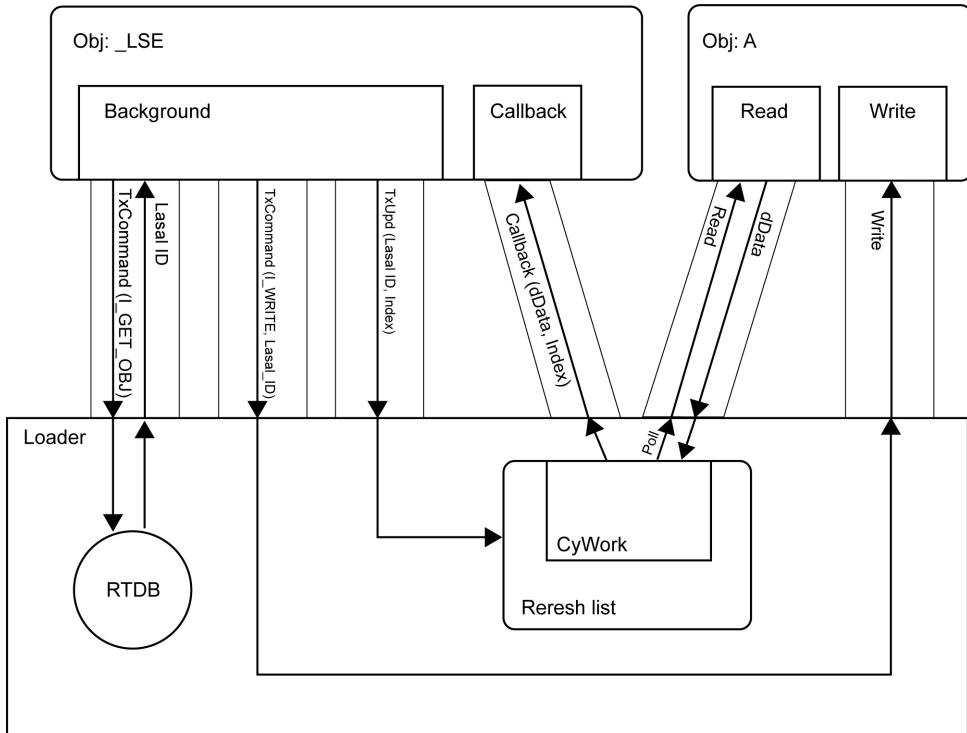
- Bilder + Ausgänge laden
- Aufbau einer dynamischen Liste

1.1.3 Comlink-Schnittstelle

Die Visualisierung ruft die Server-Werte über die Comlink-Schnittstelle ab. Die zum Aufruf der Comlink-Befehle verwendeten API-Funktionen sind im Loader implementiert.

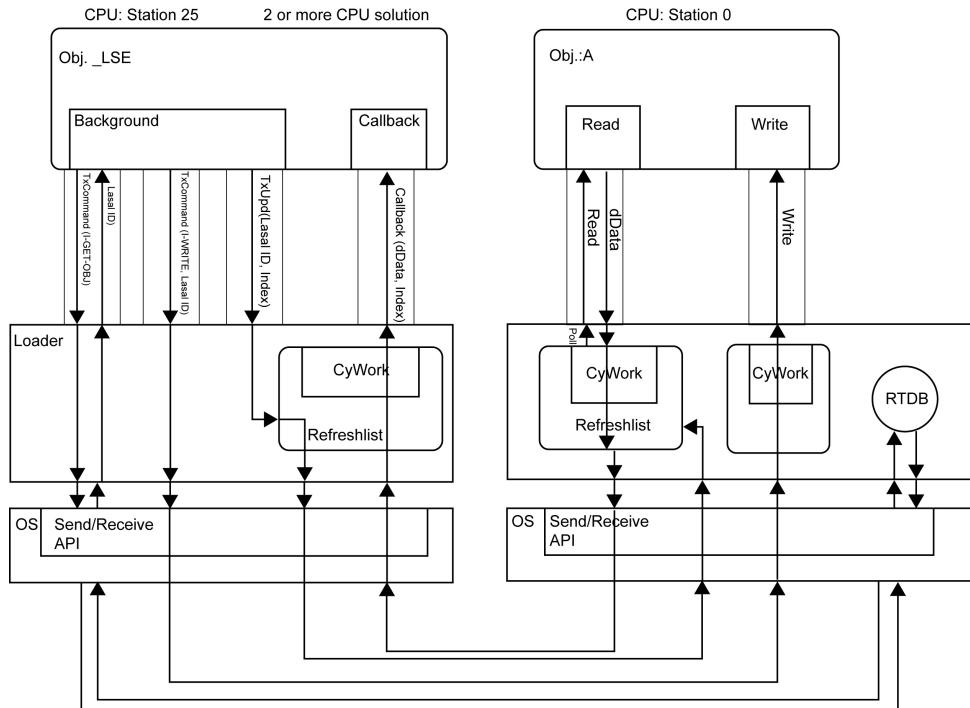
Ein Comlink-Befehl sendet eine Anfrage an eine Zielstation, die eine eigene CPU (1-CPU-Lösung) sein kann, oder auf eine CPU, die über CAN oder TC/IP verbunden ist (Mehr-CPU-Lösung).

Comlink interface 1 CPU solution



(* Grafik Comlink-Schnittstelle 1-CPU-Lösung *)

Comlink Interface



(* Grafik Comlink-Schnittstelle Mehr-CPU-Lösung; zwei oder mehr *)

Der LSE-Kernel adressiert die Zielstation über eine logische Stationsnummer. Die Zuordnung der logischen Stationsnummern für physikalische Adressen wird in der C:\IPC.INI-Datei gespeichert.

Um einen Server-Wert zu erhalten, sind folgende Comlink-Befehle erforderlich:

- Mit TxCommand wird die LasalID über einen Server-Name abgefragt.

Der Server-Wert kann direkt über die LasalID abgefragt werden (Polling), oder ein Server kann in der Refresh-Liste eingetragen werden. Wenn ein Server in der Refresh-Liste eingetragen wird, wird der Wert des Servers in regelmäßigen Abständen abgefragt und der Wert in der Gegenstation gespeichert. Es gibt zwei Refresh-Listen: Eine statische und eine dynamische. Die statische Liste wird vom LSE für Server verwendet, deren Werte ständig benötigt werden (z.B. Zeit). Bei der Änderung von Bildschirmen wird die dynamische Refresh-Liste regelmäßig abgefragt. Diese enthält die Server für den aktiven Bildschirm.

Mit der Funktion TxCommand kann eine Anweisung gesendet werden, die mit einer Lese/Schreibmethode auf eine lokale oder eine Remote-Station zugreift. Bei einer lokalen Station wird die Methode im gleichen Task ohne Warteschleife aufgerufen. Mit dem LSE-Kernel wird der Background-Task verwendet.

Für die Remote-Stationen wird der Befehl an den Loader der Zielstation übergeben. Da der zyklische Teil des Loaders im zyklischen Task läuft, wird die Methode vom zyklischen Task aufgerufen.

1.1.4 Aktualisierung des Betriebssystems

Schalten Sie die SPS nicht aus, während das System-Update läuft!



1.1.5 Backup von Persistenz-Daten

Beispielkonfiguration

Die folgenden Einträge werden in der AUTOEXEC.LSL-Datei benötigt:

```
SET RAM FORMAT NEW
```

Dateisystem

Dieses Kapitel enthält Informationen über das Dateisystem, wie die Zuweisung von Laufwerk-Zeichen und einen Überblick der Verzeichnisstruktur einer SPS, die ein Datei-System enthält.

1.1.6 Zuweisung von Laufwerk-Zeichen

LasalOS ordnet die Laufwerk-Zeichen wie folgt zu:

IPC, C-IPC

A	Erstes normales Floppy-Laufwerk
B	OS Version < 5.42: Zweites normales Floppy-Laufwerk OS Version ≥ 5.42: USB-Floppy-Laufwerk

C, D, ... Die Laufwerke der Festplatten werden in folgender Reihenfolge zugeordnet

- Alle Partitionen des Master-Geräts auf dem primären IDE-Controller.
- Nur für einen IPC: Das interne SmartMedia-Laufwerk.
- Alle Partitionen des Slave-Geräts auf dem primären IDE-Controller.
- USB-Laufwerk. Die ersten 4 Laufwerks-Buchstaben sind für vier Geräte mit einer logischer Einheit (LUN) reserviert, die nächsten zwei Laufwerks-Buchstaben sind für ein Gerät mit zwei LUNs reserviert und die nächsten vier Laufwerks-Buchstaben sind für ein Gerät mit vier LUNs reserviert.

Wenn das interne SmartMedia-Laufwerk (nur auf einem IPC) oder das Slave-Gerät auf dem primären IDE-Controller nicht physikalisch vorhanden ist, wird ein Laufwerks-Buchstabe für diese Geräte reserviert. Bei einem C-IPC werden keine Laufwerks-Buchstaben für ein SmartMedia-Laufwerk reserviert.

ET261, ET321

A-B	Nicht zugeordnet
C	RAM-Laufwerk
D	SmartMedia-Laufwerk

Beispiel

IPC mit einer Partition auf der Festplatte des primären IDE-Masters und ein USB-Laufwerk mit einer logischen Einheit:

C	Primäre Partition auf der Festplatte
D	Reserviert für den SmartMedia-Laufwerk
E	Reserviert für eine Partition der primären IDE-Slave-Disk
F	USB-Laufwerk

IPC mit einer Partition auf der Festplatte des primären IDE-Masters, einer Partition auf der Festplatte des primären IDE-Slaves und ein USB-Gerät mit zwei logischen Einheiten:

C	Primäre Partition auf der primären IDE Master-Festplatte
D	Reserviert für den SmartMedia-Laufwerk
D	Primäre Partition auf der primären IDE Slave-Festplatte

F	Reserviert für das 1· USB-Laufwerk mit einer LUN
G	Reserviert für das 1· USB-Laufwerk mit einer LUN
H	Reserviert für das 3· USB-Laufwerk mit einer LUN
I	Reserviert für das 4· USB-Laufwerk mit einer LUN
J	USB-Gerät, 1· logische Einheit
K	USB-Gerät, 2· logische Einheit

IPC mit 2 Partitionen (ein primäres Laufwerk und ein logisches Laufwerk in einer erweiterten Partition) auf der Festplatte des primären IDE-Masters und ein USB-Gerät mit vier logischen Einheiten:

C	Primäre Partition auf der Festplatte
D	Logisches Laufwerk in der erweiterten Partition der Festplatte
E	Reserviert für das interne SmartMedia-Laufwerk
F	Reserviert für eine Partition der primären IDE-Slave-Disk
G	Reserviert für das 1· USB-Laufwerk mit einer LUN
H	Reserviert für das 2· USB-Laufwerk mit einer LUN
I	Reserviert für das 3· USB-Laufwerk mit einer LUN
J	Reserviert für das 4· USB-Laufwerk mit einer LUN
K	Reserviert für das USB-Gerät mit 2 LUNs, 1· logische Einheit
L	Reserviert für das USB-Gerät mit 2 LUNs, 2· logischen Einheiten
M	USB-Gerät, 1· logische Einheit
N	USB-Gerät, 2· logische Einheiten
O	USB-Gerät, 3· logische Einheiten
P	USB-Gerät, 4· logische Einheiten

C-IPC mit 2 Compact Flash-Karten, jede Karte hat eine Partition. Ein USB-Floppy wird verbunden:

B	USB-Floppy-Laufwerk
---	---------------------

- | | |
|---|------------------------|
| C | 1: Compact Flash-Karte |
| D | 2: Compact Flash-Karte |

Übersicht des Verzeichnisbaums

Dieses Kapitel beschreibt die Systemdateien und wichtige Teile des LasalOS Verzeichnisbaums.

Systemdateien

C:\AUTOEXEC.LSL	Es handelt sich um eine Batch-Datei, die beim Start ausgeführt wird. Sie enthält typischerweise Befehle für die Konfiguration des Systems (z. B. Online-Parameter, Maus- und Touch-Einstellungen) und Befehle zum Laden und Starten der Applikation.
C:\LSLSYS	Das LSLSYS-Verzeichnis enthält Dateien, die Teil des Betriebssystems sind, z. B. DLLs, cli_help.txt und cli.sml.
C:\LSLCMD	Das Verzeichnis enthält LSLCMD-Befehlsdateien, z. B. Batch-Dateien oder Skripte.
C:\SYSMSG	Dies ist der Standard-Pfad, in dem Memo-Dateien gespeichert werden.

Projektdatei

Alle Dateien, die Teil der Applikation sind, werden im Hauptverzeichnis der Applikation gespeichert. Wenn nicht anderenfalls angegeben, ist das Hauptverzeichnis der Applikation C:\.

<appl-root>\ACTIVE.DAT	Diese Datei existiert nur auf Plattformen, bei denen die Persistenz Daten nicht in einem statischen RAM gespeichert sind. Die Datei enthält eine Kopie den Persistenz Daten und wird erstellt, wenn die Applikation beendet wird oder wenn die USV einen Stromausfall erkennt. Beim Start liest das Betriebssystem diese Datei aus und kopiert die Daten in den RAM-Bereich, wo konstante Daten gespeichert werden. ACTIVE.SAV ist eine Sicherung einer früheren ACTIVE.DAT. Hinweis: Konstante Daten werden nicht in diese Datei kopiert, wenn die USV nicht aktiviert ist und die Stromversorgung ausgeschaltet ist!
<appl-root>\ACTIVE.SAV	
<appl-root>\SRAM.DAT	Diese Datei hat den selben Zweck wie ACTIVE.DAT. Der Unterschied zwischen den beiden Dateien ist das interne Datenformat. Das Speichern von konstanten Daten ist mit SRAM.DAT viel schneller als mit ACTIVE.DAT. Daher wird eher SRAM.DAT als ACTIVE.DAT empfohlen. Um SRAM.DAT zu verwenden, muss der SET-RAM FORMAT NEW-Befehl in die AUTOEXEC.LSL-Datei eingefügt werden. SRAM.DAT wird erst ab LasalOS-Version 5.28 unterstützt und das Projekt wird mit LASAL CLASS ab Version 0.60 unterstützt.
<appl-root>\SRAM.SAV	
<appl-root>\LSLWORK	Das LSLWORK-Verzeichnis enthält die Objektdateien (LOB-Dateien) der Applikation und eine Beschreibung des Projekts sowie seiner Dateien (IDX-Datei).

<appl-root>\IPC.INI	Die IPC.INI-Datei ist eine Konfigurationsdatei für den LSE-Kernel.
<appl-root>\MPC	Das MPC-Verzeichnis enthält Dateien wie z.B. MPC-Dateien, Fonts und Bitmaps, die vom LSE-Kernel verwendet werden.

1.1.7 Protokollierung von System- und Applikationsnachrichten

Die Sysmsg-Funktion ermöglicht der Applikation Fehler, Warnungen und andere Meldungen zu erzeugen. Oft ist es wichtig, dass diese Nachrichten in eine Datei geschrieben werden, um später abgerufen werden zu können. Betriebssystem-Nachrichten und Meldungen einer Applikation werden in verschiedenen Dateien gespeichert, die zur Erkennung von Problemen angezeigt werden können.

Wird eine Meldung erzeugt, wird diese nicht sofort in eine Datei geschrieben. Stattdessen wird sie zuerst in einen Puffer geschrieben. Der Nachrichtenpuffer des Betriebssystems wird bei der Erkennung eines Power-Downs oder beim Auftritt eines schweren Fehlers in eine Datei geschrieben. Der Applikationspuffer wird beim Beenden der Applikation oder beim Erkennen eines Power-Downs in eine Datei geschrieben.

Es ist wichtig, die USV bei der Verwendung der sysmsg-Funktion zu aktivieren. Wird die USV nicht aktiviert, hat das Betriebssystem keine Chance, den Nachrichtenpuffer in eine Datei zu schreiben, wenn ein Power-Down erfolgt. Wird die USV nicht aktiviert oder ist keine USV vorhanden, wird der Nachrichtenpuffer beim nächsten Anschalten in eine Datei geschrieben. Eine Begrenzung, Dateiquote genannt, kann für die Nachrichtendatei definiert werden, um das Vollschreiben der Festplatte zu vermeiden. Wenn die Nachrichtengröße die Dateiquote überschreitet, wird die Nachricht in eine Backup-Datei umbenannt und eine neue Nachrichtendatei erstellt. Ist dies der Fall, geht eine bereits vorhandene Backup-Datei verloren. Die benötigte Größe der Festplatte für die Memo-Dateien ist daher $2 * \text{Dateiquote}$; der Dateiname ist EVENTxx.LOG, in der xx eine eindeutige Nummer für jedes Nachrichten-Pufferobjekt ist (die Betriebssystem-Nachricht-Puffernummer ist 00). Der Pfad, wo die Nachrichtendatei gespeichert wird, kann mit den Umgebungsvariablen SYMSGPATH (Betriebssystem-Nachrichten) und APPMSGPATH (Applikationsnachrichten) konfiguriert werden. Der Standardpfad ist C:\SYMSG.

Die sysmsg-Funktion kann mit den Command Line Interface (CLI)-Befehlen und den Parametern in der API-Schnittstelle Sysmsg-Funktion konfiguriert werden. Der Pfad, in dem die Nachricht gespeichert wird und die Dateiquote kann ebenfalls konfiguriert werden. Wenn ein System mehreren Disk-Dateien hat, ist es besser, den Pfad der Nachrichtendateien auf ein Laufwerk zu setzen, welches das Betriebssystem nicht enthält.

Die zur Konfiguration von sysmsg benötigten CLI-Befehle sind EventLog ON | OFF <file-quota>, SETENV SYMSGPATH <Pfad> und SETENV APPMSGPATH <Pfad>. Detaillierte Beschreibungen der CLI-Befehle finden Sie im Command Line Interface.

1.1.8 Betriebssystem-Nachrichten

Betriebssystem-Nachrichten werden erzeugt, wenn besondere Ereignisse eintreten. Jedes Ereignis hat einen Zeitstempel am Anfang der Zeile, die das Datum zeigt

04/10/01;

formatiert als YY/MM/DD (Jahr/Monat/Tag), gefolgt von einem Semikolon und die Zeit

13:32:00.692;

formatiert als hh/mm/SS/MS (Stunden/Minuten/Sekunden/Millisekunden), gefolgt von einem Semikolon

Der Zeitstempel sieht dann wie folgt aus:

04/10/01;13:32:00.692; ...

Ereignisse, die mehr Informationen protokollieren müssen enthalten den aktuellen Zeitstempel (kann später geändert werden) nur in der ersten Zeile.

Die Log-Ereignisse, die durch Interrupts ausgelöst werden, haben ein anderes Format:

*I*01:DIAS Error (3F)

Die Zeile beginnt mit

* I *

und das zeigt einen Interrupt, gefolgt von einer Zahl, die bei jeder Protokollierung eines Events um eine Zahl inkrementiert wird. Hinter dieser Zahl steht ein Doppelpunkt und der Fehler-Text.

Manche Events brauchen einen Stack-Dump, um protokolliert werden zu können. Diese Zeilen sind in der Regel 16 Hexadezimalwerte, die jeweils ein Byte darstellen. Wenn der Stack-Pointer auf einen Speicherbereich zeigt, zu dem der Zugang verweigert wird, ist kein sinnvoller Wert verfügbar und die Dump-Anzeige "???" wird für jeden Wert angezeigt.

Nicht alle protokollierten Events sind für den Anwender von Bedeutung!
Einige sind nur für die Entwickler vom LasalOS nützlich.



Eine detaillierte Liste der Events und ihre Beschreibungen:

- Erklärung der Zeitstempel wurde immer weggelassen
- Beispiele gibt es nur, wenn diese notwendig sind (Text variiert)

System-Meldungen

PowerOn, OS [I] ([III])	[I] OS-Version [II] OS build date Jedes Mal, wenn die SPS eingeschaltet wird Beispiel: 04/10/01;13:32:00.692;PowerON,OS:5.80 (Aug 12 2004, 12:55:00)
Reboot	Jedes Mal, wenn die SPS neu gestartet wird
USV Power-Down erkannt	Jedes Mal, wenn die SPS einen Power-Down-Zustand erkennt
USV Power down beendet	Power-Down-Vorgang ist beendet
Battery low!	Batterie-low-Interrupt aufgetreten
Stromausfall!	Stromausfall-Interrupt aufgetreten
Übertemperatur	Übertemperatur-Interrupt aufgetreten
Temperatursensor Fehler!	Temperatursensor Fehler-Interrupt aufgetreten
Task [I] wird beendet!	[I] Task-Name Tritt auf, wenn ein System-Task aufgrund einer Ausnahme beendet wird. Beispiel: Task TCP_SERVER0 wird beendet!

Fehlermeldungen

System

SYSKERNEL_SP: ST_MailBox full	Service-Provider Mailbox ist voll
SYSKERNEL_SP_Q ueue: nicht initialisiert	Service-Provider Mailbox (noch) nicht initialisiert
SYSKERNEL_SP_Q ueue: Queue voll	Service-Provider Befehlqueue ist voll
TaskStopHook: SPQ nicht leer	Service-Provider Befehlqueue ist nicht leer

OS Downloaden Fehler	Fehler bei der Vorbereitung des LasalOS-Downloads
WATCHDOG Fehler	CPU-Watchdog abgelaufen
F A T A L S Y S T E M F E H L E R	An diesem Punkt wird das LasalOS gestoppt. Es gibt weitere Zeilen mit ausführlichen Informationen dazu.

Download OS

▪ Nr. MEM Access	Fehler bei der Vorbereitung des LasalOS-Downloads (CPU 386)
▪ Nr. EPROM	Adresse des EPROMs nicht gefunden
▪ Image-Fehler	Länge von OS Image falsch
▪ MAPMem EPROM-Fehler	Zugriffsrechte des EPROMs können nicht eingestellt werden
▪ OS Schreibfehler	Fehler beim Schreiben des OS Images auf das EPROM
▪ Nr. RTB-Datei	Keine RTB Datei gefunden
▪ RTB-Datei ungültig	RTB Datei ungültig
▪ RTB-Filelength ungültig	RTB Dateilänge ungültig
▪ RTB-FileOpen	Fehler beim Öffnen der RTB-Datei
▪ RTB-FileCopy	Fehler beim Schreiben der RTB Datei auf die Festplatte
▪ BOOTDisk Operation	BOOTDISK-Vorgang ist fehlgeschlagen, Festplatte nicht bootfähig

Debugger

Debug_Handler: Exit (unbekannt bpnnum); ip_addr = [I]	[I] EIP of breakpoint Tritt auf, wenn der Debugger einen ungültigen Breakpoint erkennt Beispiel:
----------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------

```
Debug_Handler:Exit(unknownbpnum); ip_addr= 0x001a0fe4
```

Runtime Error

**RUNTIME
Error;Task= [I] ;Object= [II]**

[I] taskname
[II] Objektname

Wird geschrieben, wenn ein Runtime-Fehler im Real-Time, zyklischen oder Background-Task aufgetreten ist. "Taskname" und "ObjectName" definieren der Task und das Objekt, der die Fehlermeldung verursacht hat.

Application heap Error

AppMemError

Verschiedene Informationen

TCP/IP

IP-Adressen-Konflikt; [I] bereits von einer anderen Station im Netzwerk verwendet; Reaktionszeit [II]

[I] IP-Adresse
[II] Reaktionszeit

Tritt beim Initialisieren einer Ethernet-Schnittstelle mit einer IP-Adresse auf, die im Netz bereits existiert

Fehler xn_set_ip: [I]

[I] Fehler-Code

Die Einstellung einer IP-Adresse war nicht erfolgreich

Exception

Exception in AppHeap(EIP:[I]), [II]

[I] EIP vom Fehler
[II] Exception Typ

Eine Exception in einer Funktion ist aufgetreten, die den Applikations-Heap verwaltet; die Applikation wird beendet.

Exception in den aktiven Funktionen (EIP: [I]), [II]

[I] EIP vom Fehler
[II] Exception Typ

Eine Exception beim Schreiben ACTIVE.DAT / SRAM.DAT. Applikation wird beendet.

Exception [I], EIP = [II]

[I] Exception-Typ
[II] EIP vom Fehler

Eine Exception ist aufgetreten; Applikation wird beendet.

DIAS Fehler

DIAS Fehler ([I])	[I] DIAS Station (hexadezimal) Ein DIAS-Fehler ist aufgetreten
DIAS Risc Fehler ([I])	[I] DIAS Station (hexadezimal) Ein DIAS RISC-Fehler ist aufgetreten (illegale Anweisung, illegaler Befehl)

Bei beiden Fehler - "Exception-" und "DIAS-Fehler" – stehen nach den Fehlern der Informationsblöcke:

Projektname	[I]:CpStat=[II] [I] Projektname [II] Projektzustand
Tipp bei aufgetretenem Exception- oder DIAS-Fehler	[I] - Dump: [I] Exception oder DIAS
Register Dump	EIP:... EAX:... EBX:... ECX:... EDX:... ESI:... EDI:... ESP:... EBP:... EFL:... CS:... DS:... SS:... ES:... FS:... GS:...
Task, in dem ein Fehler aufgetreten ist	TASK=[I],STACK=[II]-[III],MIN=[IV] [I] Task name [II] Stack start [III] Stack end [IV] Erste nicht verwendete Stack-Adresse
Memory-Daten	APPHEAP:Start=[I],Size=[II],Used=[III],Free=[IV] [I] Applikation-Heap Start-Adresse [II] App. Größe des Heap-Speichers [III] App. verwendeter Heap-Speicher [IV] App. Heap freier Speicher USRCODE:Start=[I],Size=[II] [I] User-Code Startadresse [II] User-Code Größe USRDATA:Start=[I],Size=[II] [I] User-Daten Startadresse [II] User-Daten Größe
Stack-Dump	Beispiel:

	<p>STACK:</p> <p>00DFFE80:CA A2 15 00 86 8A 0C 00 05 00 00 00 9C FE DF 00 00DFFE90:F0 A2 15 00 10 89 0C 00 05 00 00 00 C0 FE DF 00 00DFFEA0:8A 4A 17 00 10 89 0C 00 7C FF DF 00 88 FF DF 00 00DFFEB0:30 1A 04 00 60 F9 DF 00 10 89 0C 00 70 FF DF 00</p>
Task-Liste	<p>Liste aller Aufgaben</p> <p>Beispiel:</p> <p>Haupt-Task :00D00000-00DFFF88,1033360d</p> <p>Name des Tasks</p> <p>Task Stack Start-Adresse</p> <p>Task Stack Zieladresse</p> <p>Gebrauchter Stack Bytes</p>
Call Stack	<p>Listet den Call-Stack auf. Funktionen des LasalOS haben keine Namen, sie sind mit ihrer EIP als "EIP = ..." aufgelistet. Anwendungen oder Loader-Funktionen werden mit ihrem Namen und ihrem Modul aufgelistet.</p> <p>Beispiel:</p> <p>CALL STACK:</p> <p>03792028:CLASSPRJ@.\cppcode\cppObjects\T_main.obj::_RTK_READY (00541) 03792048:CLASSPRJ@._Lse_\Lse_00_00.st::_BACKGROUND@_\LSE (02010) 03792058:EIP=0x001AD57A 03792090:EIP=0x0017D952 037920C4:EIP=0x0014A55D</p>

Debug-Meldungen

Diese Nachrichten können mit CLI-Kommandos konfiguriert werden

SET DBGLEVEL topic level

Topic

CANLL	Niedrige CAN-Routinen
APPHEAP	Applikation-Heap
GRAPHIC	Grafikbibliothek
SYSHANDLER	Exception handle
CAN	CAN-Kommunikation
CDIAS	CDIAS-Bibliothek

TASKLISTS	Task-Liste
BREAKPOINTS	Breakpoint-Handling
FLASH	CIPC Flash-Modul
IP	TCP/IP-Kommunikation
TGRAPH	LARS-Grafiken
SERIAL	Serialle Schnittstellebibliothek
USB	USB-Schnittstellenbibliothek (Standard-Level = 2)
DEBUGIP	Loader: Debug Interpreter
MODLOAD	Modul-Loader
KERNEL	LasalOS Kernel (Maintask, Serviceprovider)
LINKER	Linker-Bibliothek
LOADER	Loader-Bibliothek
TASKS	Task Handling
VNCCLI	VNC client
VNCSVR	VNC server
LSLFILE	LslFile-Bibliothek
LSLCMD	LslCmd-Bibliothek

Level

0	Keine Protokollierung
1	Fehlerprotokollierung
2	Zusätzliche Protokollierung von Debug-Informationen
3	Erweiterte Protokollierung von Debug-Informationen

Beispielkonfiguration

Betriebssystem (C:\) auf einer Compact-Flash-Karte, einer Festplatte, auf der die Nachrichten gespeichert werden sollen (D:\) und einer maximalen Betriebssystem-Datei von 2 MB. Die Betriebssystem-Nachrichten sollen in D:\SYSMSG und die Applikationsnachrichten in D:\APPMMSG gespeichert werden.

Die folgenden Einträge werden die in der AUTOEXEC.LSL Datei benötigt:

```
SET EVENTLOG ON 2000000
SETENV SYSMSGPATH D:\SYSMSG
SETENV APPMSGPATH D:\APPMMSG
```

1.1.9 Runtime-Check und Watchdog

LASAL hat folgende Runtime-Checks durchgeführt:

- peripherer CDIAS Reset
- peripherer DIAS Slave Reset
- Intelligenter Master-Watchdog
- Cyclic Runtime-Check
- Background Runtime-Check (ab LasalOS 01.01.008)
- Realtime Runtime-Check (ab LasalOS 01.01.008)
- Fehlererkennung

Dies gilt für alle CPUs (CIPC, CCL911, ...).

1.1.10 Hardware Watchdogs

Es gibt drei Hardware Watchdogs, die die Peripheriegeräte überwachen. Werden sie nicht rechtzeitig ausgelöst, werden die Peripheriegeräte zurückgesetzt. Es gibt zwei Phasen, in denen die Watchdogs ausgelöst werden:

Init phase	In dieser Phase werden die Hardware und Hardware-Klassen initialisiert. Die Watchdogs müssen für die Initialisierung getriggert werden; dies erfolgt durch das Betriebssystem
RunRam phase	Hier wird die Hardware komplett initialisiert und die Applikation steuert die Geräte. Hier lösen die Hardware-Klassen den Watchdog aus.

Die Watchdogs werden in den anderen Phasen (Power-up, Booten des Betriebssystems, Status rücksetzen, ...) nicht ausgelöst. Die Hardware-Watchdog-Zeit läuft alle 130 ms aus, die Software-Trigger sind in Schritten von 100 ms eingestellt. Dieser Wert wurde gewählt, um zusätzliche Zeit zur Verfügung zu stellen und somit Software-Jitter zu ermöglichen, jedoch kann der Wert an Benutzeranforderungen angepasst werden.

peripherer CDIAS Reset	Das CDIAS-Gerät enthält eine Watchdog-Schaltung, die ein Peripheriegerät-Reset auslöst, wenn die Watchdog-Schaltung innerhalb von 130 ms nicht ausgelöst wird. Der CDIAS-Watchdog wird durch das Betriebssystem in der Init-Phase und die Hardware-Klassen im RunRam-Status ausgelöst.
DIAS Peripherie-Slave-Reset	Ein DIAS Slave-Gerät enthält einen Watchdog-Zähler, der ein Reset auslöst, wenn nicht innerhalb von 130 ms auf den Bus zugegriffen wird. Der Watchdog wird durch das Betriebssystem in der Init-Phase und die Hardware-Klassen im RunRam-Status ausgelöst.

**Intelligenter Master-
Watchdog**

Der intelligente Master enthält einen Watchdog-Zähler, der den intelligenten Master ausschaltet, wenn kein Zugriff auf ein spezielles Register innerhalb von 130 ms erfolgt. Der Watchdog wird durch das Betriebssystem in der "Init"-Phase und die Hardware-Klassen im RunRam-Status ausgelöst.

1.1.11 Software-Tests

Zusätzlich zu den Hardware-Watchdogs gibt es mehrere Runtime-Checks, die den Realtime-, Cyclic- und Background-Task überwachen. Wenn die Laufzeit für einen Task (die gesamte Laufzeit der jeweiligen Methoden) einen voreingestellten Wert überschreitet, stoppt die SPS die Applikation und setzt das Peripheriegerät zurück (weil kein Watchdog getriggert wird).

Cyclic Runtime-Check

Das Betriebssystem scannt die Liste der Objekte ständig mit einem CyWork Methode und ruft die CyWork Methode auf, wenn die Call-Zeit der Methode abgelaufen ist. Wird beim Scannen dieser Liste die vordefinierten Zeit überschritten, wird das Programm gestoppt und ein RUNTIME (CPU Status Code 2) Fehler ausgelöst.

Der Cyclic-Runtime-Check kann wie folgt eingestellt werden:

1. CLI-Befehl "SET RUNTIME <time in 10ms units>".
2. Für CPUs ohne ein CLI (Command Line Interface) enthält die globale Variable `_swruntime` den Wert des zyklischen Laufzeit-Checks in 10 ms-Einheiten

Ein Wert von 0 deaktiviert den Cyclic-Runtime-Check.

Background Runtime-Check

Das Betriebssystem scannt die Liste der Objekte ständig mit einer CyWork-Methode und ruft die Background-Methode auf, wenn die Call-Zeit der Methode abgelaufen ist. Wird beim Scannen dieser Liste die vordefinierte Zeit überschritten, wird das Programm gestoppt und ein BT-RUNTIME Fehler ausgelöst (CPU-Status-Code 68).

Der Background Runtime-Check kann wie folgt eingestellt werden:

1. CLI-Befehl "SET BTRUNTIME <time in 10 ms Einheiten>".
2. Für CPUs ohne CLI (Command Line Interface) enthält die globale Variable `_sbtruntime` den Wert des zyklischen Laufzeit-Checks in 10 ms-Einheiten.

Ein Wert von 0 deaktiviert den Background Runtime-Check.

Realtime Runtime-Check

Wenn die Dauer aller RtWork-Methoden länger als das Realtime-Intervall ist, wird das Programm gestoppt und ein RT-RUNTIME Fehler (CPU-Status-Code 67) ausgelöst. Die Dauer des Realtime-Intervalls kann mit dem CLI-Befehl SET RTRUNTIME eingestellt werden.

1.1.12 Fehlererkennung

Wenn das LasalOS einen Fehler erkennt, wird die Applikation beendet (es gibt derzeit eine Exception, den "DIAS Error").

DIAS Error

Das CLI-Kommando SET DIASERROR OFF, deaktiviert diesen Fehler. Wurde "DIAS Error" mit dem CLI-Kommando SET DIASERROR ON aktiviert, beendet die Standardeinstellung der SPS die Applikation und setzt die Peripheriegeräte zurück.

Das "OS_SSR_InstallDIASHandler"-Makro erlaubt der Applikation einen "DIAS Error"-Handler zu installieren, der aufgerufen wird, wenn "DIAS Error" aktiviert ist und ein Fehler auftritt. Liefert der Handler "0" (Null) zurück, wird kein Fehlerzustand erzeugt und die Applikation wird weiter ausgeführt. In allen anderen Fällen wird die Applikation gestoppt und die Peripheriegeräte werden zurückgesetzt.

Ein DIAS Error kann drei Mögliche Ursachen haben:

- DIAS allgemeine Störung (Zugang zum Gerät nicht vorhanden)
- DIAS RISC Fehler
- DIAS unbekannter Opcode-Fehler

Anderen Fehlerbedingungen

Die Anwendung wird beendet und die Peripheriegeräte werden mit folgenden Fehlern zurückgesetzt:

1. CPU Exceptions
 - Divide-Fehler, Overflow, ungültiger Opcode, Coprozessor nicht verfügbar, Doppelfehler, Coprozessor Segment-Überschreitung, ungültiges TSS, Segment nicht vorhanden, Stack-Exception, Zugang-Exception, Seitenfehler;
2. LOADER Error
 - Wenn der Loader einen Fehler erkennt (kein Speicherplatz, unbekannte Klassen-ID, unbekannter Konstruktor, unbekanntes Objekt, unbekannter Kanal, falscher Anschluss, falsches Attribut, Syntaxfehler, virtueller Dateifehler, Klasse nicht kompatibel, Klasse muss aktualisiert werden), bleibt der Zustand unverändert (kein

Umschalten in den Reset-Zustand, weil ACTIVE.DAT mit falschen Daten geschrieben werden würde), aber die Watchdogs werden nicht mehr ausgelöst. Deshalb werden die Peripheriegeräte zurückgesetzt. Realtime-, zyklisch- und Background-Task laufen in diesem Zustand nicht.

3. HEAP Error

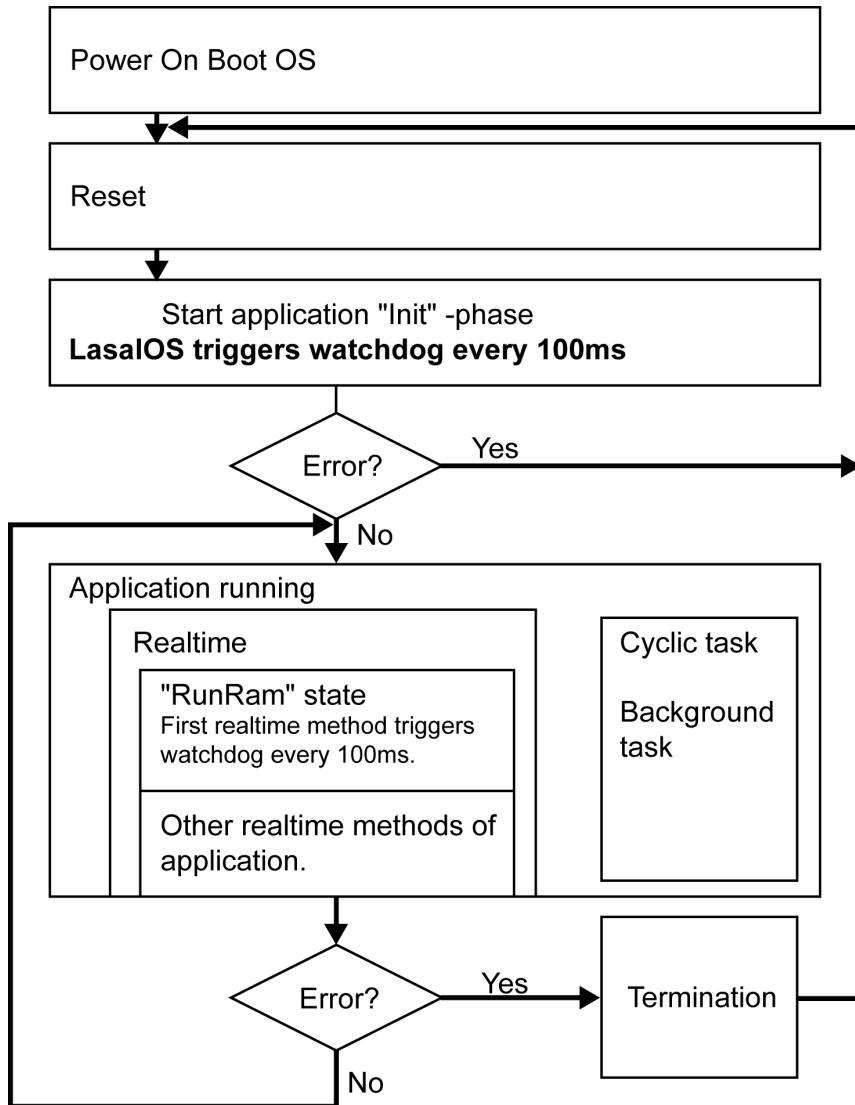
- "APPLMEM ERROR" beendet die Applikation, Watchdogs werden nicht ausgelöst, Peripheriegeräte werden zurückgesetzt.

4. Fehler aufgrund von optionalen Debug-Einrichtungen

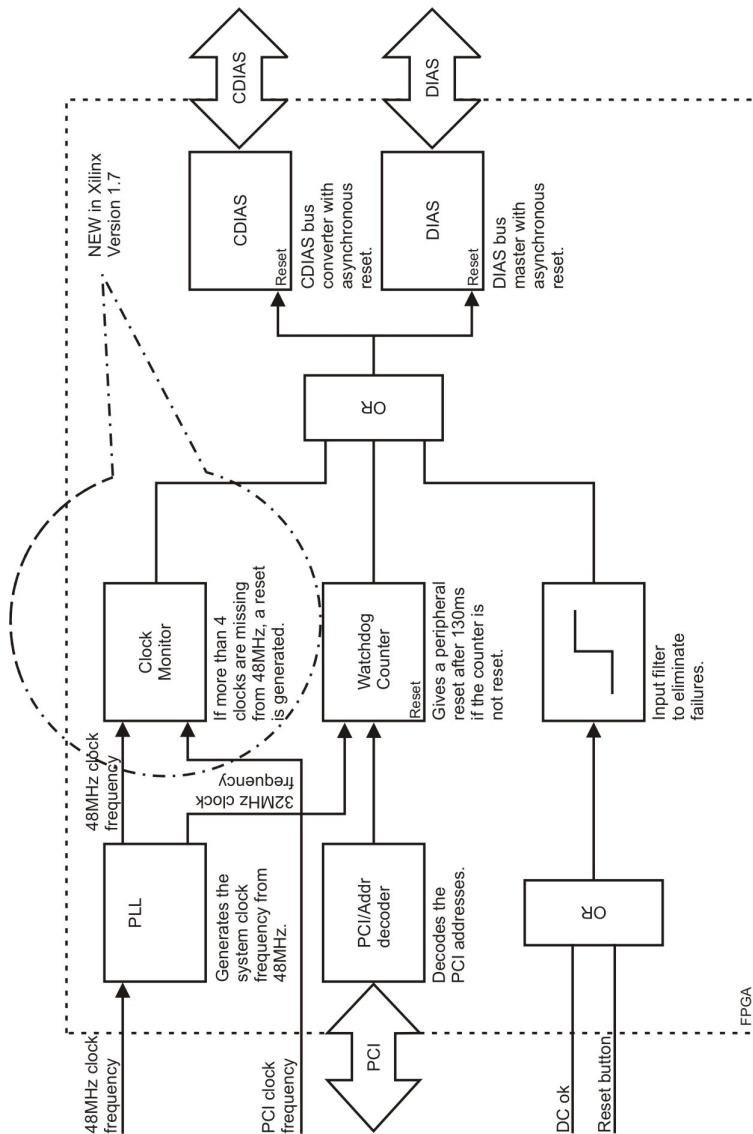
- HEAP

- Mit Hilfe des CLI-Befehls "SET DBGHEAP ALL ON" kann man zusätzliche Bedingungen zur Erstellung eines "APPLMEM ERROR" mit mehr Details im Event-Log erzeugen.
- Die Applikation wird beendet, der Watchdog wird nicht ausgelöst, Peripheriegeräte werden zurückgesetzt.

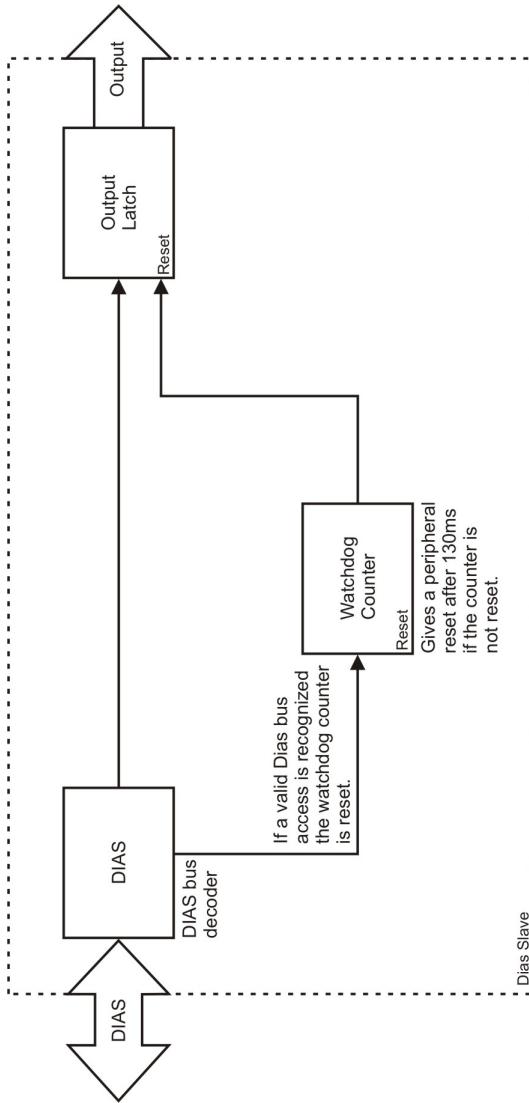
1.1.13 Operationsablauf



1.1.14 CIPC Peripherie-Reset



1.1.15 Peripherer DIAS Slave Reset



1.1.16 Einfacher Router

Um einen IPC oder eine CPU als einen einfachen Router zu verwenden, müssen zwei Netzwerkschnittstellen vorhanden sein. Die beiden Netzwerkschnittstellen dürfen sich nicht im gleichen Logiknetzwerk befinden, was bedeutet, dass die IP-Adressen der Schnittstellenverbindungen so konfiguriert werden müssen, dass die erste nicht die zweiten erreichen kann.

Eine solche Konfiguration ist jedoch nicht möglich.

Bei der Verwendung der autoexec.lsl für das Einstellen der IP-Adressen entsteht ein Konflikt mit der zweiten Adresse. Das hat zur Folge, dass LASAL OS die zweite Schnittstelle nicht installiert.

Das Command Line Interface zeigt eine Meldung auf dem Bildschirm (falls vorhanden).

Beispiel

LASAL CLASS Online-Verbindung von einem Windows-PC auf einen IPC über einen einfachen CIPC Router.

Windows PC

IP Adresse	10100100100
Subnet	255.255.255.0
Gateway	10.100.100.150

Der Windows-PC muss über ein Gateway verfügen. Wird ein Paket über das Netzwerk geschickt und kein anderer Host (ein anderer PC im gleichen Netzwerk) kann auf das Paket zugreifen, wird es an ein Gateway geschickt.

Das Gateway (Router) ist ein zugreifbarer Host im Netzwerk mit einer zweiten Netzwerkschnittstelle, um das Paket weiterzuleiten.

Die obige Erklärung dient nur, um ein Verständnis davon zu geben, wie sich der Router verhält. Der Windows-PC weiß, dass kein Host auf das Paket zugreifen kann und schickt es direkt an das Gateway.

CIPC

2 Netzwerkschnittstelle ("Router/Gateway")

Netzwerkschnittstelle 1	autoexec.lsl Einstellungen	
IP-Adresse	10100100150	SET IP HOSTADDR 10 100 100 150

Subnet	255.255.255.0	SET IP SUBNET 255 255 255 0
Gateway	nein	SET IP GATEWAY 255 255 255 255

Netzwerkschnittstelle 2		autoexec.lsl Einstellungen
IP-Adresse	192.168.43.155	SET IP 2 HOSTADDR 192 168 43 155
Subnet	255.255.0.0	SET IP 2 SUBNET 255 255 0 0
Gateway	nein	SET IP 2 GATEWAY 255 255 255 255

IPC

autoexec.lsl Einstellungen		
IP-Adresse	192.168.43.154	SET IP HOSTADDR 192 168 43 154
Subnet	255.255.0.0	SET IP SUBNET 255 255 0 0
Gateway	192.168.43.155	SET IP GATEWAY 192 168 43 155

Mit dieser Konfiguration ist es möglich, einen PC mit der IP-Adresse 10.100.100.100 zu einem IPC in einem anderen Netz mit der IP-Adresse 192.168.43.154 zu verbinden.

Der Windows-PC sendet ein Paket an 192.168.43.154. Diese Adresse darf sich nicht im gleichen logischen Netzwerk befinden wie der Windows-PC.

Das Paket wird an das Gateway 10.100.100.150 gesendet. Diese Adresse ist die erste Schnittstelle des einfachen Routers. Die Netzwerkschnittstelle 1 leitet intern das Paket an die zweite Netzwerkschnittstelle mit der Adresse 192.168.43.155 weiter.

Die zweite Netzwerkschnittstelle sendet das Paket dann an 192.168.43.154.

1.1.17 Downgrade Sicherheit

Sie können diese Funktion verwenden, um zu verhindern, dass per USB Stick eine falsche (alte) Betriebssystemversion eingespielt wird.

Downgrade Sicherheit aktivieren

- den Eintrag min_update_version=x in der Datei C:\lslsys\config.lsl hinzufügen
- wobei x der Versionsnummer entspricht. (min_update_version=2)
- am USB-Stick eine Datei namens version.lsl erstellen

- den Eintrag has_version=2 in diese Datei einfügen.
- nur wenn has_version größer oder gleich min_update_version ist, wird das Skript am USB-Stick ausgeführt

1.1.18 Von LASALOS unterstützte USB-Geräte

Diese Treiber sind für alle CPUs mit Ausnahme von CCL722, DCL642 und DTC081 verfügbar.

Hub-Treiber	Alle Standard-Hubs werden unterstützt.
Tastatur-Treiber	Der Tastatur-Treiber unterstützt alle USB-Tastaturen mit USB-Klasse 3 (HID), Subclass 1 (Boot Device) und des Protokolls Nr. 1 (Tastatur). Bis zu vier Tastaturen können gleichzeitig betrieben werden. Zusätzliche Tastaturen werden ignoriert.
Maus-Treiber	Der Maus-Treiber unterstützt alle USB-Mäuse mit USB-Klasse 3 (HID), Subclass 1 (Boot Device) und des Protokolls Nr. 2 (Maus). Bis zu vier Mäuse können gleichzeitig betrieben werden. Weitere Mäuse werden ignoriert. Eingeschlossen sind auch HAMPSHIRE Octopus-Touch-Controller.
Druckertreiber	Der Druckertreiber unterstützt alle USB-Drucker mit USB-Klasse 7 (Drucker), Subclass 1 oder 2 (uni- oder bidirektional), und jedes Protokoll. Bis zu vier Drucker können gleichzeitig betrieben werden. Weitere Drucker werden ignoriert.
Disk-Treiber	Der USB-Disk-Driver unterstützt alle USB-Disks, die den USB Mass Storage Device Spezifikation, Code Klasse 8, Subclass 4 (UFI), 5 (FDD) oder 6 (SCSI), Protokolle 0 (Control/Bulk/Interrupt), 1 (Control/Bulk/Interrupt, kein Interrupt-Status) oder 0x50 (Bulk only) entsprechen. Es wird eine beliebige Anzahl von Geräten mit jeweils bis zu 2 Terabyte Kapazität unterstützt. Auch Geräte mit integrierten Hubs bzw. mehrere LUNs werden unterstützt.
USB to Serial	Bis zu vier Geräte "ATEN UC232A" werden unterstützt. Sie stehen als COM7-COM10 zur Verfügung.
Barcode-Leser	Die Barcode-Leser CipherLab 1067, CipherLab 1090 und CipherLab 1021 mit HID-Schnittstelle werden unterstützt. Sie werden wie Tastaturen behandelt; die gelesenen Daten sind als Tastenkette vorhanden. Außer dem Eingabefeld ist keine spezielle Software erforderlich.

1.1.19 Von LASALOS unterstützte USB-Geräte für CCL722, DCL642 und DTC081

Disk-Treiber

Der USB-Disk-Driver unterstützt alle USB-Disks, die den USB Mass Storage Device Spezifikation, Code Klasse 8, Subclass 4 (UFI), 5 (FDD) oder 6 (SCSI), Protokolle 0 (Control/Bulk/Interrupt), 1 (Control/Bulk/Interrupt, kein Interrupt-Status) oder 0x50 (Bulk only) entsprechen. Es wird nur ein Laufwerk unterstützt. Der Drive darf keinen eingebetteten Hub enthalten. Enthält den Drive mehrere LUNs, wird nur die erste unterstützt.

USB-to-Ethernet-Treiber

Der USB-to-Ethernet-Treiber unterstützt alle USB-to-Ethernet-Adapter mit dem folgenden Chipsatz:

- ASIX AS88772
- ASIX AS88172

Die folgenden Produkte wurden geprüft:

- SiIG, Inc - USB 2.0 zu 10/100 Ethernet, Modellnummer US2277
- D-Link - USB 2.0 Fast-Ethernet-Adapter, Modellnummer DUB-E100

Nicht auf allen Plattformen verfügbar.



1.1.20 USB-Update

Anforderungen

Der USB-Stick wird mit der CPU direkt (nicht über einen HUB) verbunden und sollte nicht über einen integrierten Hub verfügen. Bei Steuerungen mit einem RTK-Betriebssystem darf während des Updatevorgang nur ein USB-Stick, nämlich der Updatestick, an der Steuerung angeschlossen sein.

1. Projekt-Update

Um ein Projekt-Update durchzuführen, muss eine Bootdiskette auf einem USB-Stick erstellt werden.

Die folgenden Schritte sind durchzuführen:

- Stecken Sie den USB-Stick an den PC
- Laden Sie das Projekt in LASAL CLASS
- Bauen Sie das Projekt wieder auf
- Klicken Sie auf "Create Bootdisk ..." im Projektmenü.
Wählen Sie im erscheinenden Dialog den USB-Drive aus und klicken Sie auf OK.

Außerdem müssen Sie mit den folgenden Kommandos eine autoexec.lsl-Datei im Hauptverzeichnis des Sticks erstellen:

```
LSLOAD F:  
LSLSAVE
```

Nun können Sie den USB-Stick mit der CPU verbinden und einen Neustart durchführen.
Während des Boot-Verfahrens wird das Projekt in den internen Flash geladen.

2. Update LSE-Projekt

Um das LSE-Projekt zu aktualisieren, müssen Sie alle Dateien aus dem Runtime-Verzeichnis des LSE-Projekts auf den Stick kopieren und einige Dateioperationen in der autoexec.lsl durchführen:

```
MKDIR C:\MPC           REM Verzeichnis erstellen, wenn noch nicht vorhanden  
XCOPY F:\LSE\*.* C:\MPC\  REM Projekt-Dateien kopieren
```

Es stehen auch andere Dateioperationen zur Verfügung: DEL, rmdir, REN und FORMAT.



Das C: Laufwerk auf der CPU ist der gepufferte Flash-Disk-Drive; das F: Laufwerk ist der USB-Drive.

3. OS-Update

Um ein OS-Update durchzuführen, ist folgender Befehl in autoexec.lsl notwendig:

```
BOOT F:\<file name>  
<Dateiname> ist der Name der BIN-Datei (z.B. CCL722.BIN) auf dem USB-Stick
```

Schliessen Sie den Stick an der CPU und führen Sie ein Neustart aus. Während der Boot-Operationen wird das Betriebssystem aktualisiert. Die Aktualisierung ist abgeschlossen, wenn die CPU in den Zustand "OS installiert" schaltet. Trennen Sie den USB-Stick und starten Sie das neue OS noch einmal neu.

Beispiel autoexec.lsl für CCL722

```
rem Project Update
LSLLOAD F:\Project
LSLSAVE

rem Copy LSE Files
MKDIR C:\MPC
DEL C:\MPC\*.*
XCOPY F:\LSE\*.* C:\MPC\

rem OS Update
BOOT F:\OS\ccl722.bin
```

4. Update "autoexec.lsl"

Um eine Aktualisierung der autoexec.lsl durchzuführen, muss eine bestehende autoexec.lsl-Datei von einem USB-Stick in das Hauptverzeichnis der CPU (C:\) kopiert werden.

Folgender Befehl ist notwendig:

```
XCOPY <source> <destination>
```

Befindet sich die autoexec.lsl, die vom USB-Stick ausgeführt wird, im gleichen Verzeichnis wie die autoexec.lsl, die in dem Hauptverzeichnis von der CPU kopiert werden soll, muss ein anderer Name verwendet werden.

```
XCOPY F:\autoexec.tmp C:\autoexec.lsl      REM same dir., different name
XCOPY F:\TEMP\autoexec.lsl C:\autoexec.lsl  REM different dir., same name
```

5. Update von HTML-Dateien

Webseiten können von der CPU in einen Browser übertragen werden und über den Browser durch einen verfügbaren Web-Server auf der CPU dargestellt werden.

Dies ist nicht auf allen Plattformen verfügbar!



Die Dokumentation [LASALOS Web-Server](#) enthält eine detaillierte Beschreibung der Konfiguration und Nutzung des Web-Servers.

Um ein Update der HTML-Dateien durchzuführen, müssen alle Dateien in die CPU kopiert werden. Alle Verzeichnisse und Unterverzeichnisse müssen vorhanden sein (müssen erstellt werden, falls sie nicht existieren).

```
MD HTML
MD HTML\Images
XCOPY F:\HTML-Files\* C:\HTML
XCOPY F:\HTML-Files\Images\* C:\HTML\Images
```

Der Befehl XCOPY unterstützt Wildcards (*, **, *.txt, ...), aber er funktioniert nicht rekursiv (ohne Unterverzeichnis).

6. Unterstützung von langsamen USB-Sticks

Die Initialisierungszeiten von USB-Sticks können sich je nach Hersteller stark unterscheiden.

Daher kann es vorkommen, dass während dem Hochlauf der SPS der USB-Stick am USB nicht bzw. zu spät erkannt wird, obwohl dieser am USB-Port angesteckt ist.

Dies kann dann dazu führen, dass die am USB-Stick hinterlegte Datei autoexec.lsl beim Starten nicht gefunden und somit auch nicht ausgeführt wird.

Um solche USB-Sticks dennoch zu unterstützen, kann die Hochlaufzeit um eine konfigurierbare Zeit erhöht werden.

Hierfür ist in der Datei C:\lslsys\config.lsl der Eintrag bootStickDetectionDelayTime=x einzufügen, wobei x der Zeit in Millisekunden entspricht, um die die Hochlaufzeit verlängert wird. Der maximal zulässige Wert für den Eintrag beträgt 15000 ms.

Die Hochlaufzeit wird durch den oben angeführten Eintrag generell erhöht, unabhängig davon, ob tatsächlich ein USB-Stick angesteckt ist.

1.1.21 Setup/Defragmentierung Compact-Flash

Installieren eines LasalOS

Um eine LasalOS-Version auf einem Compact-Flash installieren zu können, benötigen Sie das Windows-Tool bootdisk.exe und die RTB oder LBI des gewünschten Betriebssystems. Beide sind bei Bedarf vom SIGMATEK-Support erhältlich. Führen Sie in der Command-Prompt den folgenden Befehl aus.

```
Bootdisk os.ext x:
```

os	Dateiname des gewünschten Betriebssystems
----	-------------------------------------------

ext	RTB or LBI
x:	Laufwerks-Buchstabe des Compact Flashs

Formatierung einer Compact-Flash

In bestimmten Fällen, z.B. wenn alle Dateien gelöscht werden sollen, kann es nötig sein, die Compact-Flash zu formatieren. Das Compact-Flash muss mit mindestens 4 Sektoren pro Cluster korrekt formatiert werden.

Dies erfolgt automatisch durch die Format-Anweisung in LasalOS. Um eine Compact-Flash mit Windows zu formatieren, müssen Sie

Format / A: 2048 / FS : FAT-X :
am command-prompt eingeben.

x:	Laufwerks-Buchstabe des Compact Flashs
----	----------------------------------------

Defragmentierung einer Compact-Flash

Ein Compact-Flash-Medium kann mit einem Windows-Betriebssystem defragmentiert werden. Nach der Defragmentierung ist es notwendig, das LasalOS wie oben beschrieben neu zu installieren.

1.1.22 Lasal OS Tasks

Task-Namen	Kurze Info
Main Task	Task zur Bearbeitung des ServiceProvider-Queues: Befehle zum Starten und Beenden der Applikation durch die Online-Verbindung und durch die OS_SSR_AddToKernelSP der Applikation
Idle Task	idle task
CPU Monitor	idle-Task (berechnet die CPU-Zeit)
DLED_Task	Task zur Aktualisierung der LEDs
RT_AsyncTask	Task für die asynchronen Datei-Methoden der Klasse _FileSys
PANELTASK	Task für die Tastatur, LEDs und das Command Line Interface (DTC281)
CMD_LINE	Command Line Interface
KEY_TASK	Task für die Tastatur (CET281, BDF2000)
LCD_TASK	Refresh-Task für LCD

CANx_Stationx	CAN-Online-Verbindung
CanLL Timer	Timer-Task für das CAN-Protokoll
CanRxMailThread	Task für alle ankommenden CAN-Nachrichten
CANx_Listener	Aufgabe für alle ankommenden CAN-Nachrichten (LARS)
PLC386_TASK	Um die Baudrate, ... während des Starts durch die SET-Taste einzustellen; einstellen der LEDs, Applikation laden
Execute Task	zur Ausführung aller Funktionen durch das EXEC-Kommando, das mit der Funktion OS_SSR_AddToKernelSP aufgerufen wird
HiPrio Task	Zum Stoppen der Applikation aufgrund von Fehlern in einer Interrupt-Routine (DIAS Error)
UserLogTask	Task zum asynchronem Schreiben der Eventlog-Dateien
KEYLED_TASK	Task für die Tastatur und LEDs (z. B. ET261)
RT_Runtime	Hoher Prioritäts-Task, um die Applikation aufgrund eines Realtime-Runtime Errors zu stoppen
TCP Client x	TCP Online-Verbindung
TCP Server	TCP-Server für die Online-Verbindung
IP TASK	Interner Task des IP-Stacks (pro Netzwerkschnittstelle)
INTERRUPT	Interner Task des IP-Stacks (pro Netzwerkschnittstelle)
TIMER	Interner Task des IP-Stacks
DbgRxFast	Serielle Online-Verbindung
TASK_DBGSERWIN 32	Serielle Online-Verbindung (LARS)
SSFDCTask	Task für den SmartMedia-Treiber
USB Hub	USB hub
USB Mouse	USB mouse
USB Keyboard	USB keyboard
USB Touch	USB touch
USB Print	USB printer
KEYSHUTTLE_TASK	Task zur Tastatur und das Shuttle-Rad
VNC SERVER	VNC server
WSP0_CT	cyclic task
WSP0_RT	realtime task

WSP0_BT	background task
---------	-----------------

1.1.23 Anlegen von CAN-Identifiers

1.1.23.1 SPS (Online + Comlink Protokoll)

Die folgenden CAN-Identifier Bereiche für das Comlink-Protokoll:

- 16#700 – 16#71F
- 16#500 – 16#67F (vorausgesetzt, dass der Standardwert für die maximalen Comlink-Kanäle nicht verändert wird)

Die Identifier der CAN-Objekte, die tatsächlich angelegt werden, hängen von folgenden Faktoren ab:

- CAN-Stations-Nummer der SPS (STATION), Bereich: 0-31
- Anzahl den ausgehenden Comlink-Verbindungen (NBR_CONNECTIONS)
- Max. Anzahl der Comlink-Kanäle (MAX_CHANNELS). Der Standardwert ist 5. Dieser Wert kann angepasst werden, indem Sie im Loader (Lasal1: loader.h, Lasal2: UserDef.h) die #define COMLINK_CAN_COMCHS ändern. Nachdem dieser Wert geändert wurde, muss der Loader kompiliert und mit der Anwendung verlinkt werden. Es ist darauf zu achten, dass alle Steuerungen, die über das Comlink-Protokoll kommunizieren, den gleichen Wert verwenden!

Die folgenden Formeln braucht man zur Berechnung der zugeteilten Identifiers:

```

a) 16#700 + STATION
b) IF MAX_CHANNELS < 8 THEN
    From: 16#500 + STATION * (MAX_CHANNELS + 1) * 2
    Count: NBR_CONNECTION * 2
ELSE
    From: 16#200 + STATION * (MAX_CHANNELS + 1) * 2
    Count: NBR_CONNECTION * 2
END_IF

```

Beispiel

3 SPS mit den Stationsnummern 0, 11, 25.

Stationen 11 und 25 stellen eine Verbindung zur Comlink-Station 0 her.

Der Standardwert für COMLINK_CAN_COMCHS bleibt unverändert (-> MAX_CHANNELS = 5).

Station 0

```
STATION = 16#00, NBR_CONNECTION = 0, MAX_CHANNELS = 5
```

```
16#700 + 16#00 = 16#700
From: 16#500 + STATION * (MAX_CHANNELS + 1) * 2 = 16#500
Count: 0
```

Station 11

```
STATION = 16#0B, NBR_CONNECTION = 1, MAX_CHANNELS = 5
16#700 + 16#0B = 16#70B
From: 16#500 + STATION * (MAX_CHANNELS + 1) * 2 = 16#584
Count: 2
```

Station 25

```
STATION = 16#19, NBR_CONNECTION = 1, MAX_CHANNELS = 5
16#700 + 16#19 = 16#719
From: 16#500 + STATION * (MAX_CHANNELS + 1) * 2 = 16#62C
Count: 2
```

-> Angelegte CAN-Objekte: 16#700, 16#70B, 16#719, 16#584, 16#585, 16#62C, 16#62D

1.1.23.2 MiniDisplay

Eine Instanz der Klasse MiniDisplay verwendet auch die definierten CAN-Identifiers, abhängig von der vom Benutzer konfigurierten Identifikation (über die Client NodeNumber).

Empfangsobjekt	16#180 + n
Versenden von Objekten	16#200 + n, 16#300 + n, 16#400 + n, 16#500 + n

wobei n die Stationskennung angibt.

1.1.24 Aufzählen von Ethernet-Schnittstellen-Verbindungen

Da eine SPS mehrere Ethernet-Schnittstellen unterstützen kann, wurde vereinbart, diese nach einem festen Schema zu beziffern.

1	erste interne Schnittstelle (ETH1)
2	Ethernet-Schnittstelle bei der EWP (oder ETH2 in der CCL912)
3	VARAN-Schnittstelle
4-6	reserviert
7	USB zum Ethernet-Adapter

8 zweite USB zum Ethernet-Adapter

1.1.25 Status- und Fehlermeldungen

Während des Status-Tests der LASAL CLASS-Software werden Status- und Fehlermeldungen ausgegeben. Hat das Gerät eine CPU mit einer Status-Anzeige, werden die Status- und Fehlermeldungen ebenfalls hier angezeigt. POINTER- und CHKSUM-Fehler werden auch auf dem Bildschirm des Terminals angezeigt.

Neben der Anzeige von "Er" und einem Code von 0 bis 255 können folgende Kombinationen auftreten:

Pr Ln	derzeit verbunden
Er Ln	Verknüpfungsfehler
Er Ld	Projekt wird geladen
Er OP	OPSYS programmiert derzeit
Er AL	Fehlerursache / Status unbekannt
Er Lo	Loader-Fehler. Es gibt keine weiteren Informationen über die Ursache des Fehlers. Die Applikation hat einen Fehler (unbekanntes Objekt, unbekannte Verbindung, etc.).

Nummer	Meldung	Bedeutung	Ursache/Abhilfe
00	RUN RAM	Das Anwenderprogramm läuft derzeit im RAM. Das Display wird nicht beeinflusst.	
01	RUN ROM	Das im Memory-Modul gespeicherte Anwenderprogramm wurde in den RAM geladen und läuft derzeit. Das Display wird nicht beeinflusst.	
02	RUNTIME	Die Gesamtdauer aller zyklischen Objekte überschreitet die maximale Zeit; die Zeit lässt sich mit 2-System-Variablen einstellen: <ul style="list-style-type: none"> Runtime: Verbleibende Restzeit SWRuntime: Voreingestellter Wert für den Runtime-Zähler 	
03	POINTER	Falsche Pointer wurden vor der Ausführung des Anwenderprogramms erkannt.	Mögliche Ursachen <ul style="list-style-type: none"> Memory-Modul fehlt, ist nicht programmiert oder ist beschädigt.

			<ul style="list-style-type: none">• Programm im User-Programm-Speicher (RAM) ist nicht ausführbar.• Der Batteriepuffer ist fehlgeschlagen.• Ein Software-Fehler, aufgrund dessen das Anwendungsprogramm überschrieben wird. <p>Abhilfe</p> <ul style="list-style-type: none">• Das Memory-Modul neu programmieren. Wenn der Fehler weiterhin besteht, tauschen Sie das Modul aus.• Die Pufferbatterie austauschen.• - Programmfehler beheben
04	CHKSUM	Es wurde eine falsche Prüfsumme vor der Ausführung des Anwendungsprogramms festgestellt.	Ursache/Lösung: Siehe POINTER
05	WATCHDOG	Das Programm wurde durch die Watchdog-Logik abgebrochen.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none">• Anwendungsprogramm-Interrupts werden über einen langen Zeitraum blockiert (STI-Befehl vergessen).• Ein Hardware-Interrupt wurde falsch programmiert.• INB-, OUTB-, INW-, OUTW-Befehle wurden falsch verwendet.• Prozessor ist beschädigt. <p>Abhilfe</p> <ul style="list-style-type: none">• Programmfehler beheben• Zentraleinheit austauschen
06	GENERAL ERROR	GENERAL ERROR	
07	PROM DEFECT	Ein Fehler ist beim Programmieren des Memory-Modul aufgetreten.	<p>Mögliche Ursachen:</p> <ul style="list-style-type: none">• Das Memory-Modul ist beschädigt

			<ul style="list-style-type: none"> Das Anwenderprogramm ist zu groß Das Memory-Modul fehlt <p>Abhilfe</p> <ul style="list-style-type: none"> Memory-Modul austauschen
08	RESET	Die CPU hat einen RESET-Befehl empfangen und wartet auf weitere Befehle. Das Anwenderprogramm wird nicht bearbeitet.	
09	WD DEFECT	Die Watchdog-Logik ist defekt. Nach dem Start kontrolliert die CPU die Watchdog-Funktionen. Erscheint eine Fehlermeldung während der Prüfung, wird die CPU absichtlich in eine Endlosschleife gebracht und übernimmt keine weiteren Befehle.	<p>Abhilfe</p> <ul style="list-style-type: none"> Zentraleinheit austauschen
10	STOP		
11	PROG BUSYS		
12	PROGRAM LENGTH		
13	PROG END	Das Speicher-Modul wurde erfolgreich programmiert.	
14	PROG MEMO	Die CPU programmiert gerade das Programmspeichermodul.	
15	STOP BRKPT	Das Programm wurde via Breakpoint gestoppt.	
16	CPU STOP	Die CPU wurde von der PG-Software unterbrochen (F6 HALT beim Status-Test).	
17	INT ERROR	Die CPU hat das falsche Interrupt ausgelöst und das Anwenderprogramm gestoppt oder ein unbekanntes Kommando wurde während des Programmablaufs erkannt.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none"> Ein Betriebssystem-Befehl wurde verwendet, der nicht existiert. Stack-Fehler (ungleichen Anzahl von PUSH und POP-Befehle). Anwenderprogramm durch einen Software-Fehler unterbrochen. <p>Abhilfe</p>

			<ul style="list-style-type: none"> • Programmfehler beheben
18	SINGLE STEP	CPU ist im Single-Step Modus und wartet auf weitere Befehle.	
19	READY	Ein Modul oder Projekt wurde an die CPU übertragen und ist bereit, das Programm zu starten.	
20	LOAD	Das Programm wird gestoppt, während die CPU ein Modul oder ein Projekt empfängt.	
21	UNZUL. MODUL	Die CPU hat ein Modul, das nicht zum Programm gehört.	
22	MEMORY FULL	Der Betriebssystem-Speicher (Heap) ist zu klein. Es konnte kein Speicher durch den Aufruf einer internen bzw. Schnittstellenfunktion aus der Applikation reserviert werden.	
23	NOT LINKED	Ein fehlendes Modul oder ein Modul, das nicht zum Projekt gehört wurde während des CPU-Starts festgestellt.	
24	DIV BY 0	Bei einer Division ist ein Fehler aufgetreten.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none"> • Division durch 0 • Der Quotient passt nicht in das Ergebnisregister. <p>Abhilfe</p> <ul style="list-style-type: none"> • Programmfehler beheben
25	DIAS ERROR	Ein Fehler ist beim Zugreifen auf ein DIAS-Modul aufgetreten.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none"> • Es wurde versucht, auf ein nicht vorhandenes DIAS-Modul zuzugreifen • DIAS-Bus-Fehler <p>Abhilfe</p> <ul style="list-style-type: none"> • DIAS-Bus kontrollieren • Abschlusswiderstände kontrollieren
26	WAIT	Die CPU ist beschäftigt.	
27	OP PROG	Betriebssystem wird neu programmiert.	
28	OP INSTALLED	Betriebssystem ist neu installiert.	

29	OS TOO LONG	Das Betriebssystem kann aufgrund des unzureichenden Speicherplatzes nicht geladen werden.	
30	NO OPERATING SYSTEM	Bootloadermeldung Kein Betriebssystem im RAM gefunden.	
31	SEARCH FOR OS	Der Bootloader sucht im RAM nach dem Betriebssystem.	
32	NO DEVICE		
33	UNUSED CODE		
34	MEM ERROR	Das installierte Betriebssystem wurde nicht für die Hardware entwickelt.	
35	MAX IO		
36	MODULE LOAD ERROR	Das Modul oder LASAL-Projekt konnte nicht geladen werden.	
37	GENERAL OS ERROR	Allgemeiner Fehler beim Laden des Betriebssystems.	
38	APPLMEM ERROR	Fehler in der Anwendung dynamische Speicher-Administration (Benutzerverwaltungs-Heap).	
39	OFFLINE		
40	APPL LOAD		
41	APPL SAVE		
46	APPL-LOAD-ERROR	Fehler beim Laden der Applikation.	
47	APPL-SAVE-ERROR	Fehler beim Speichern der Applikation.	
50	ACCESS-EXCEPTION-ERROR	Lese-/Schreib-Zugang auf beschränkten Speicherbereich, wie z. B.: auf NULL-Pointer schreiben.	
51	BOUND EXCEEDED	Exception-Fehler beim Überschreiben des Speicherbereichs.	
52	PRIVILEGED INSTRUCTION	Unerlaubter Befehl für aktuelle CPU-Ebene, z. B. Einstellung der Segmentregister	
53	FLOATING POINT ERROR	Fehler bei einer Fließkomma-Operation.	
60	DIAS-RISC-ERROR	Fehler von intelligenten DIAS Master.	

64	INTERNAL ERROR	Interner Fehler - alle Applikationen wurden gestoppt.	Neustarten, Nachricht an SIGMATEK schicken
65	FILE ERROR	Fehler bei einer Dateioperation	
66	DEBUG ASSERTION FAILED	Interner Fehler	Neustarten, Nachricht an SIGMATEK schicken
67	REALTIME RUNTIME	Die Gesamtdauer aller Realtime-Objekte überschreitet die maximale Zeit; dieser Wert kann nicht kann nicht konfiguriert werden: 2 ms bei 386er CPUs 1 ms bei allen anderen CPUs	Ab Version 1.1.7
68	BACKGROUND RUNTIME	Die Gesamtzeit aller Background-Objekte überschreitet die maximale Zeit; die Zeit kann mit 2 System-Variablen konfiguriert werden: <ul style="list-style-type: none">• BTRuntime: verbleibende Zeit• SWBTRuntime: vorausgewählter Wert für die Runtime-Zähler	
70	C-DIAS ERROR	Es ist ein Fehlerfall in Verbindung mit einem C-DIAS-Modul aufgetreten.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none">• Die Ursache dieses Fehlers ist im Logfile dokumentiert <p>Abhilfe</p> <ul style="list-style-type: none">• Das kommt auf die Ursache an
72	S-DIAS ERROR	Es ist ein Fehlerfall in Verbindung mit einem S-DIAS-Modul aufgetreten.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none">• reales Netzwerk stimmt nicht mit Projekt überein• S-DIAS Client ist defekt <p>Abhilfe</p> <ul style="list-style-type: none">• Logfile auswerten
75	SRAM-FEHLER	Es ist ein Fehler beim Initialisieren, Lesen oder Schreiben der SRam-Daten aufgetreten.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none">• SRam falsch konfiguriert• SD-Karte falsch formatiert• SD-Karte entfernt• Batterie für die Versorgung des internen Programmspeichers ist leer

			<p>Abhilfe</p> <ul style="list-style-type: none"> • Log-File auswerten (Event00.log) • Konfiguration überprüfen • Batterie für die Versorgung des internen Programmspeichers wechseln • SD-Karte mit LASAL CLASS 2 als EDGE-Medium formatieren • SD-Karte überprüfen
95	USER DEFINED 0	Benutzerdefinierbarer Code	
96	USER DEFINED 1	Benutzerdefinierbarer Code	
97	USER DEFINED 2	Benutzerdefinierbarer Code	
98	USER DEFINED 3	Benutzerdefinierbarer Code	
99	USER DEFINED 4	Benutzerdefinierbarer Code	
100	C_INIT	Beginn der Initialisierung, Konfiguration wird verarbeitet.	
101	C_RUNRAM	Das LASAL-Projekt wurde erfolgreich vom RAM gestartet.	
102	C_RUNROM	Das LASAL Projekt wurde erfolgreich vom ROM gestartet.	
103	C_RUNTIME		
104	C_READY	Alles ist in Ordnung	
105	C_OK	Alles ist in Ordnung	
106	C_UNKNOWN_CID	Unbekannte Klasse eines Stand-alone- oder Embedded-Objekts oder unbekannte Basisklasse.	
107	C_UNKNOWN_CONSTR	Die Betriebssystem-Klasse kann nicht erstellt werden, vielleicht falsches Betriebssystem.	
108	C_UNKNOWN_OBJECT	Ein unbekanntes Objekt wurde in einem Interpreter-Programm gefunden. Erstellung von mehr als einem DCC080-Objekt.	

109	C_UNKNOWN_CHN L	Die Anzahl der HW-Module ist größer als 60.	
110	C_WRONG_CONNE CT	Keine Verbindung zu den benötigen Kanälen.	
111	C_WRONG_ATTR	Falsche Server-Attribute.	
112	C_SYNTAX_ERROR	Es gibt keinen spezifischen Fehler, alle alle Projektabschnitte werden neu kompiliert und erneut geladen.	
113	C_NO_FILE_OPEN	Es wurde versucht, eine unbekannte Tabelle zu öffnen.	
114	C_OUTOF_NEAR	Memory-Zuordnung ist fehlgeschlagen.	
115	C_OUTOF_FAR	Memory-Zuordnung ist fehlgeschlagen.	
116	C_INCOMPATIBLE	Ein Objekt mit dem gleichen Namen, aber einer anderen Klasse ist bereits vorhanden.	
117	C_COMPATIBLE	Ein Objekt mit dem gleichen Namen und der Klasse ist bereits vorhanden, muss aber aktualisiert werden.	
224	LINKING	Die Applikation wird derzeit verknüpft.	
225	LINKING ERROR	Fehler bei der Verknüpfung, Nachricht wird im LASAL Status-Fenster angezeigt.	
226	LINKING DONE	Verknüpfen beendet.	
230	OP BURN	Das Betriebssystem wird im Flash gespeichert.	
231	OP BURN FAIL	Fehler beim Brennen des Betriebssystems	
232	OP INSTALL	Betriebssystem wird derzeit installiert.	
240	USV-WAIT	Versorgung wurde deaktiviert, USV ist aktiv.	
241	REBOOT	Neustart des Betriebssystems.	
242	LSL SAVE		
243	LSL LOAD		
252	CONTINUE		
253	PRERUN	Applikation wird gestartet.	
254	PRERESET	Applikation wird beendet.	
255	CONNECTION BREAK		

1.1.26 Routing

1.1.26.1 Übersicht

- CPUs können miteinander vernetzt werden, so dass jede CPU als ein Gateway zu einer anderen CPU dienen kann.
- Die Adressierung erfolgt über Kennungen (0-254).
- Das Netzwerk kann als Hybrid konfiguriert werden, das bedeutet, dass es keine Rolle spielt, ob CAN oder Ethernet oder beides verwendet wird.
- Die Routing-Tabelle wird der Applikation über eine Variable zur Verfügung gestellt.
- Das Routing funktioniert nur für das Lasal32-Protokoll.
- Alias ist möglich.
- Routes müssen eindeutig sein.
- Routing über RS232 wird nicht unterstützt.
- Es kann nur eine Route über eine CAN-Leitung geben.
- Unterstützte CPUs sind: C-IPC, CCL91x, CCL722, DCL642, DTC081, DTC081_IP, ET261, ETT321, IPC und BDF2000

1.1.26.2 Kommandos

```
set station <n>
```

Hier können Sie die Station-ID für eine CPU einstellen. Die Kennung muss vor die Routing-Tabelle gesetzt werden! "set station" ohne einen zusätzlichen Parameter zeigt die aktuell eingestellte Kennung.

```
set routingtable <filename.ext>
```

Lädt die Routing-Tabelle der angegebenen Dateinamen. Nur die Einträge, die für die CPU relevant sind, werden interpretiert (von der CPU-ID). Die Routing-Tabelle wird gelöscht, wenn der Befehl ohne Parameter emittiert wird.

1.1.26.3 Routing-Tabelle Syntax

Die Datei wird Zeile für Zeile analysiert; Einträge sind nicht abhängig von der Groß- und Kleinschreibung. Falsche Einträge sowie Einträge, die nicht für die aktuelle CPU vorgesehen sind, werden ignoriert. Es kann nützlich sein, die Ausgabe dieser Befehle zu prüfen.

```
„station“<n> [,ip <TargetIP>:<Port>“|„can <TargetCAN> “ | „local“]
```

n	0-254
TargetIP	IPv4 address
Port	default Port 1954
TargetCAN:	0-31 (CAN-Stationsnummer)
Lokal	Die Zielstation ist mit der lokalen Station identisch (Alias).

„;“, „#“ und „rem“ zeigt Anmerkungen an.

Beispiel

;Station0		
STATION0	1	station10
STATION0	24	ip 10.10.116.66:1954
Station0	010	ip 10.10.116.59:1954
#Station1		
station1	0	station10
station1	10	can 10
station1	24	station10
rem Station10		
station10	0	ip 10.10.116.68:1954
station10	24	ip 10.10.116.66:1954
station10	1	can 0
;station24		
station24	0	ip1 010.010.116.68:1954
station24	1	station10
station24	10	ip 10.10.116.59:1954
#loopback		
station0	254	lokal

Jede Station entscheidet unabhängig, wie ein Paket verarbeitet wird. Wenn eine Station ein Paket mit einer unbekannten Zielstation zum Routen empfängt, wird das Paket verworfen und eine Fehlermeldung wird zurückgegeben.

1.1.27 Boot-Screen

Plattformen, die eine grafische Farboberfläche unterstützen und ein Dateisystem haben, können einen benutzerdefinierten Start-Bildschirm anstelle des CLI anzeigen.

Der benutzerdefinierte Bildschirm muss in C:\bootpic.bmo gespeichert werden, so dass die CPU ihn finden kann.

Die Datei kann mit Hilfe des bmp2bmo.exe-Tools im LASAL Screen-Editor erzeugt werden.

Das Boot-Image wird durch eine der folgenden Events geladen:

- Die Applikation startet (gibt es keine _LSE, wird der Standard-Bildschirm angezeigt)
- <TAB> oder <ESC> wird während des Bootens betätigt
- Online-Kommando wird von LASAL während des Starts empfangen
- Reset-Befehl von LASAL

Eine Activity-Bar wird am Bildschirm eingeblendet, um den Benutzer zu informieren, dass die CPU beschäftigt ist. Außerdem wird eine Meldung über der Activity-Bar angezeigt, nachdem autoexec.lsl beendet wurde.

Trace-Nachrichten werden unterdrückt. Ist eine Exception in der Applikation aufgetreten, zeigt die CPU die CLI an.

Situationen, die Benutzer berücksichtigen sollten:

Störung	Die Meldung "Autoexec.lsl beendet!" wird nicht angezeigt, aber die Activity-Bar wird ausgeführt.
Grund	Ein "PAUSE" Befehl oder eine andere Anweisung erwartet eventuell eine Benutzereingabe (diese ist natürlich unsichtbar während der Anzeige des Start-Bildes).
Abhilfe	Entfernen Sie entweder das Boot-Bild oder die interaktiven Befehle aus der autoexec.lsl.
Störung	Es wird ein statischer Bildschirm angezeigt.
Grund	Die CPU ist abgestürzt oder der Touchscreen ist nicht richtig kalibriert.
Abhilfe	Drücken Sie <ESC>; gibt es keine Antwort, kontrollieren Sie Ihre autoexec.lsl und entfernen Sie vorübergehend das Boot-Bild, um zu sehen, was während des Bootens passiert. Ist die CLI richtig, identifizieren Sie den

interaktiven Befehl z. B. "set mouse type hamusb" mit einer fehlenden Touch-Konfigurationsdatei und starten Sie manuell.

Beachten Sie, dass ein Boot-Bild nur auf Systemen verwendet werden darf, die die Entwicklungsphase überstanden haben! Eine kurzzeitige Entfernung des Boot-Bildes kann entweder durch das Umbenennen der Datei bootpic.bmp oder der Einstellung des CLI-Befehls "bootpic exit" zu einer gewünschten Stelle innerhalb der autoexec.lsl führen.

1.1.28 Dateisystem für USB-Sticks

Analysen zeigen, dass es von Vorteil sein kann, einen USB-Stick mit einem FAT32-Dateisystem anstelle eines FAT16-Dateisystems zu formatieren, um die Geschwindigkeit erheblich zu verbessern.

1.2 Task-Management

1.2.1 Threads

Im LasalOS werden Threads parallel geführt, wobei jeder Thread einen Code-Abschnitt eigenständig verarbeitet und über ein eigenes Registerabbild und einen separaten Stack verfügt. Threads werden auch Tasks genannt, sind aber nicht mit LASAL Tasks zu verwechseln (LASAL Tasks werden im folgenden Kapitel beschrieben).

Da in einem Single-Prozessorsystem mehrere Threads nicht gleichzeitig ausgeführt werden können, schaltet der Prozessor zwischen den Threads hin und her. Der Scheduler entscheidet, welcher Thread zu aktivieren ist, um sicherzustellen, dass ein ausgeführter Thread die CPU erreicht, wenn kein anderer Thread mit einer höheren Priorität vorhanden ist. Wenn der Scheduler zwischen mehreren Threads mit gleicher Priorität entscheiden muss, nimmt er den Thread, der die längste Wartezeit hat.

Der Scheduler wird aufgerufen, wenn ein Ereignis eintritt, das den Thread-Zustand verändert. Zum Beispiel ändert sich ein Thread in den Zustand Ready (ausführbarer Zustand), nachdem eine Nachricht erhalten wird. Der Scheduler wird in regelmäßigen Zeitabständen aufgerufen, die Timer-Tick heißen. Der Wert von Timer-Tick ist nicht auf allen Plattformen gleich. Der Timer-Tick gibt an, in welchem Zeitabstand die LASAL-Tasks gestartet werden. Für einen IPC beträgt dieser Wert 1 ms, für einen 386-Prozessor ist dieser Wert 2 ms. Bei aktuellen Plattformen kann der Timer-Tick eingestellt werden (Set Maintimer). Dabei sind Werte von 125 µs bis 2000 µs möglich.

Es gibt zwei Arten von Threads: Applikations- und Betriebssystem-Threads. Jede LASAL Task-Kategorie (Real-Time, Cyclic und Background) hat einen eigenen Applikations-Thread (Real-Time, Cyclic and Background).

Die Betriebssystem-Threads sind für die Initialisierung, Überwachung, Zeitverwaltung und Zuweisung von Services, sowie die Online-Kommunikation verantwortlich.

Die folgende Tabelle gibt einen Überblick über die wichtigsten Threads (Tasks) von LasalOS:

Thread (Task)	Task	Priorität
Realtime	Ausführen des RtWork LASAL Tasks	16
OS Kernel Task	Runtime-Überwachung (Cyclic-, Background-), Durchführung von Betriebssystemtasks	15
Cyclic	Ausführen des CyWork LASAL Tasks	14
Kommunikations-Task	Online-Kommunikation (Betriebssystem)	14
Background	Ausführung des LASAL Background Tasks Die Priorität der Background-Tasks hängt von der VISU LOW HIGH Einstellung (LOW: 10, HIGH: 14) ab.	10/14

1.2.2 LASAL Tasks

LASAL Tasks sind Methoden eines LASAL-Objekts (RtWork, CyWork oder Background), die vom Betriebssystem zyklisch aufgerufen werden. Wird die Anwendung initialisiert, schreibt der Loader den LASAL Task in die Task-Liste. Task-Listen sind für Real-Time, Cyclic und Background Tasks verfügbar. Jede Task-List Eintrag enthält einen Zeitabstand (Periode), der in LASAL Klasse (bzw. den Objekten) konfiguriert werden kann.

Die Applikations-Tasks werden in einer Endlosschleife ausgeführt, um jeden Eintrag (Objekt) zu prüfen, ob der Zeitabstand seit dem letzten Aufruf abgelaufen ist. Ist die angegebene Zeit abgelaufen, wird die Methode aufgerufen und der Zeitpunkt des letzten Aufrufs aktualisiert.

Nachdem die Task-Liste abgeschlossen ist (Applikation beendet / im Reset / im Error), werden die Applikationstasks für eine bestimmte Zeit weiterlaufen gelassen, damit die Task-Liste abgearbeitet werden kann. Der Real-Time-Task wird in einem Zeitraum von 1 ms in IPCs und 2 ms für 386 CPUs ausgeführt. Bei aktuellen Plattformen wird dieser Wert konfiguriert (Tick: 125 µs – 2 ms). Der Zeitraum für den Cyclic-Task liegt bei mindestens 1 ms, kann aber höher liegen, wenn der Tick höher ist. Der Background-Task liegt in der Regel bei 2 ms.

VISU LOW/HIGH

Die Visualisierung (LSE) wird im Background-Task durchgeführt. Die Visualisierung ist benachteiligt, wenn die CPU eine große Cyclic- oder Realtime Belastung hat, da dem Background standardmäßig eine geringere Priorität zugewiesen wird. Mit der VISU HIGH Einstellung kann der Background-Task auf die gleiche Priorität wie der Cyclic Task erhöht werden.

Das CLI-Kommando, SET VISU LOW | HIGH kann verwendet werden, um die Prioritätseinstellung von VISU LOW/HIGH zu ändern.

LASAL-Task Runtime:

Da die LASAL-Tasks der gleichen Kategorie sequenziell durchgeführt werden, beeinflusst die Laufzeit eines LASAL-Tasks die Aufrufzeit der anderen LASAL-Tasks innerhalb derselben Kategorie. Die Laufzeiten der LASAL-Tasks sollten so gering wie möglich gehalten werden, um die konfigurierten Zykluszeiten zu behalten.

Wenn sich ein Betriebssystemaufruf in einem LASAL-Task befindet, der den Applikations-Thread blockiert, werden die LASAL-Tasks der entsprechenden Kategorien ebenfalls blockiert. Der LASAL-Task wird nur weiter ausgeführt, wenn der Applikations-Thread wieder laufen kann. Betriebssystemaufrufe, wie auch TCP/IP-Funktionen oder Multithread-Schnittstellen-Funktionen können zur Blockierung führen. Diese Anrufe sollten in einer RtWork-Methode vermieden werden.

Steuerung und Kontrolle

Die Ausführung der LASAL-Tasks kann über die Laufzeitüberwachung sichergestellt (überwacht) werden. Konfiguriert wird diese in der autoexec.lsl oder im CLI mit folgenden Kommandos:

set rtruntime Wert	für die Realtime-Überwachung – Wert kann 0-255 enthalten und wird in Maintimer-Ticks angegeben. Die Dauer eines Maintimer-Ticks wird mit dem CLI-Befehl SET MAINTIMER eingestellt.
set runtime Wert	für die Cyclic-Überwachung – Wert kann 0-255 enthalten und wird in 10 Millisekunden-Schritten angegeben (z.B. 30 entspricht = 300 ms).
set btruntime Wert	für die Background-Überwachung – Wert kann 0-255 enthalten und wird in 10 Millisekunden-Schritten angegeben (z.B. 30 entspricht = 300 ms).

Der Wert 0 deaktiviert die Überwachung der jeweiligen Kategorie. Benötigt die Abarbeitung einer Task-Liste länger als die angegebene Zeit in der Laufzeitüberwachung, wird die Applikation gestoppt und in einen Laufzeitfehler gestellt.

Die Ausführungszeiten für LASAL-Tasks können über die folgenden Variablen überwacht werden:

- `_RealMaximumTime`
- `_RealAverageTime`
- `_CyclicMaximumTime`
- `_CyclicAverageTime`
- `_BackgroundMaximumTime`
- `_BackgroundAverageTime`

Die `MaximumTime`-Variable gibt die maximale Laufzeit für eine Task-Liste ab dem Start des Projekts (oder ab dem letzten Reset der Variablen) an. Die `AverageTime`-Variable gibt den Durchschnittswert der Laufzeit einer Task-Liste an, wobei sich der Wert aus dem letzten Zyklus mit 50 %, dem vorletzten mit 25 %, usw. errechnet. Diese Variablen werden in Mikrosekunden angegeben und sind in der `Rtos_variables.h`-Datei definiert.

Locking-Interrupts

Sperren von Interrupts können sich negativ auf die zeitlichen Eigenschaften von LASAL-Tasks auswirken. Es kann dann der Timer-Interrupt- und anschließend der Scheduler-Aufruf verzögert werden. Die Verzögerung des Timer-Interrupt-Aufrufes ändert den Zeitraum des Realtime-Tasks; dies erhöht den Jitter.

Wenn die Interrupts für mehr als 1 Timer-Tick gesperrt sind, gehen die Timer-Interrupts verloren und das Betriebssystem verzögert die interne Uhr. Die Realtime-Aufrufe gehen dann verloren und die Anwendungen enthalten die falschen Zeitwerte.

1.3 Programming Guide

1.3.1 Der Konstruktor

Der Konstruktor wird bei der Erstellung eines Objekts vom Loader angerufen und kann zur Initialisierung der Objektdaten-Elemente verwendet werden.

Es stehen zwei Konuktoren zur Verfügung:

- Standard-Konstruktor
- Benutzerdefinierter Konstruktor

Standard-Konstruktor

Der Standard-Konstruktions-Code wird automatisch generiert und stellt daher kein Problem für den Benutzer dar. Eine seiner Aufgaben ist zum Beispiel die Erstellung von Methodenlisten für virtuelle Methoden. Der Aufruf des benutzerdefinierten Konstruktors - wenn vorhanden - ist die letzte Aufgabe. Der Standard-Konstruktor ist STD.

Benutzerdefinierter Konstruktor

Der benutzerdefinierte Konstruktor kann vom End-Programmierer erstellt werden. Der Methodenname des benutzerdefinierten Konstruktors ist der gleiche wie der Klassename. Falls ein Konstruktor im folgenden Dokument angesprochen wird, ist der benutzerdefinierte Konstruktor gemeint.

Konstrukt Funktion Prototyp

```
FUNCTION NewClass0::NewClass0
VAR_OUTPUT
    ret_code: CONFSTATES;
END_VAR
```

Der Rückgabewert des Konstruktors verursacht einen Loader-Fehler, wenn der Wert nicht gleich C_OK ist und das Projekt deshalb nicht gestartet wird.

Folgendes ist bei Verwendung des Konstruktors zu beachten:

Ein Objekt kann nicht auf einen Client-Kanal zugreifen, wenn andere Objekte oder Verbindungen mit anderen Objekten existieren.

Wenn ein Client mit einer OSI (Operating System Interface) Klasse verbunden ist, kann dieser Kanal verwendet werden, da diese Verbindung vor dem Aufruf des Konstruktors erstellt wird.

Die Startwerte für die privaten Datenelemente des Objekts werden vor dem Aufruf des Konstruktors initialisiert. Wird ein privates Datenelement im Konstruktor initialisiert, wird der Startwert des Objekts überschrieben.

Betriebssystemaufrufe, die nicht über eine OSI-Klasse erfolgen, können in über ein Makro (OS_SSR_xxx) aufgerufen werden.

1.3.2 Exception Handling

Eine Exception ist ein Ereignis, das unerwartet ist oder die Fähigkeit der Anwendung normal fortzufahren beeinträchtigt. Exceptions können durch Hardware- oder Software festgestellt werden.

Die Anwendung kann einen Exception-Handler installieren, der reagieren kann, wenn ein Exception-Fehler auftritt. Was geschieht, nachdem der Handler ausgeführt wurde, hängt vom Rückgabewert ab. Der Handler kann eine der folgenden Aktionen durchführen:

- Die Kontrolle an den System-Handler übertragen (in der Regel wird die Anwendung sofort gestoppt).
Rückgabewert 0 (EXCEPTION_OSHANDLER)
- Fehler beheben und den normalen Steuerungsbetrieb weiterführen.
Rückgabewert -1 (EXCEPTION_CONTINUE_EXECUTION)
- Ausführung des aktuellen Objekts überspringen und mit der Ausführung des nächsten Objekts weiter machen.
Rückgabewert: 1 (EXCEPTION_SKIP_CONTINUE)

Die letzten beiden Optionen (-1 und 1) stoppen die Anwendung nicht und die Option 1 ist unter keinen Umständen verfügbar. Sie kann nur verwendet werden, wenn eine Exception in einer zyklischen Methode auftritt. Der Code wird in einer zyklischen Methode ausgeführt, wenn er direkt oder indirekt von einer der zyklischen Methoden (CyWork, RtWork oder Background) aufgerufen wird. Beispiele von nicht in einer zyklischen Methode verwendetem Code sind der Konstruktor eines Objekts, die Init-Methode, oder Schreib-/Lese-Methoden des Loaders oder eines Tasks, die mit OS_MT_CREATETHREAD erstellt wurden.

1.3.3 APIs

1.3.3.1 OS_SSR_SetHandler

Das OS_SSR_SetHandler ermöglicht einer Applikation eine Handler-Routine zu installieren, die im Fall einer Exception aufgerufen wird.

Macro

```
OS_SSR_SetHandler(p1,p2,p3)
```

Function prototype:

```
FUNCTION GLOBAL __cdecl P_SetHandler  
VAR_INPUT  
    handlerType    : DINT;  
    pFct          : pVoid;
```

```

param          : pVoid;
END_VAR
VAR_OUTPUT
ret0          : DINT;
END_VAR;

```

Übergabeparameter	Typ	Beschreibung
handlerType	DINT	Legt den Veranstaltungstyp fest, der zu einem Handler-Aufruf führen soll. Derzeit wird nur HANDLERTYPE_EXCEPTION unterstützt.
pFct	pVoid	<p>Ein Zeiger auf eine Handler-Funktion mit einem __cdecl-Aufruf. Der Prototyp dieser Funktion sieht wie folgt aus:</p> <pre> FUNCTION __cdecl EvalException VAR_INPUT excptCode : UDINT; excptInfo : ^EXCEPTION_POINTERS; taskType : UDINT; pThis : pVoid; param : pVoid; END_VAR VAR_OUTPUT retVal : DINT; END_VAR </pre> <p>Die Parameter sind:</p> <ul style="list-style-type: none"> excptCode: identifiziert den Typ der aufgetretenen Exception. Mögliche Werte, die vorkommen können, sind EXCEPTION_ACCESS_VIOLATION, EXCEPTION_INT_DIVIDE_BY_ZERO usw. (siehe Isl_st_ifssr.h). excptInfo: ist ein Pointer, der Informationen über die Exception enthält, z.B. Registerwerte zum Zeitpunkt der Exception. Wenn keine Informationen vorhanden sind, kann dieser Pointer NIL sein! taskType: ist der Task, in dem die Exception aufgetreten ist. Mögliche Werte sind TASK_OBJ_CT, TASK_OBJ_RT usw. (siehe Isl_stitask.h). pThis: ist ein Pointer auf das Objekt, das den Code während der Exception ausgeführt hat. Wenn keine Objektinformationen vorhanden sind, kann dieser Parameter NIL werden, (z.B. wenn ein User-Task eine Exception erzeugt). param: ist der Parameter, der an OS_SSR_SetHandler übergeben wird. retVal: definiert, was nach dem Handler passiert. Mögliche Werte sind EXCEPTION_OSHANDLER,

		EXCEPTION_CONTINUE_EXECUTION oder EXCEPTION_SKIP_CONTINUE.
param	pVoid	Dieser Wert wird an die Handler-Funktion übergeben. Das Betriebssystem interpretiert diesen Wert nicht; er wird nur übergeben und kann verwendet werden, um beliebige Informationen an den Handler zu geben.
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	

1.3.3.2 OS_SSR_SkipOverIDIV

OS_SSR_SkipOverIDIV kann von einer Handler-Routine im Falle eines Divisionsfehlers aufgerufen werden, um die Division zu überspringen und die nächste Anweisung weiter auszuführen.

Makro

OS_SSR_SkipOverIDIV(p1,p2,p3,p4)

Function prototype:

```
FUNCTION GLOBAL __cdecl P_SkipOverIDIV
VAR_INPUT
    exceptionCode : UDINT;
    exceptionInfo : pVoid;
    quotient      : UDINT;
    remainder     : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
exceptionCode	UDINT	Identifiziert den Typ der Exception. Dieser Parameterwert sollte der gleiche sein wie der Exception-Code, der an die Handler-Routine übergeben wird.
exceptionInfo	pVoid	Dies ist ein Pointer, der Informationen über die Exception enthält. Dieser Parameterwert sollte der gleiche sein wie der Exception-Informationen-Zeiger, der an die Handler-Routine übergeben wird.
quotient	UDINT	Der Wert dieses Parameters ist der Quotient, wenn die Division übersprungen wird.

remainder	UDINT	Der Wert dieses Parameters ist der, wenn die Division übersprungen wird.
Rückgabeparameter	Typ	Beschreibung
retval	DINT	Der Rückgabewert ist EXCEPTION_CONTINUE_EXECUTION, wenn die Division übersprungen wird oder im Fehlerfall EXCEPTION_OSHANDLER. Setzen Sie diesen Wert als Rückgabewert der Handler-Routine.

Beispiel

```
#include "LSL_ST_EXCPT.H"
#include "LSL_ST_IFSSR.H"

VAR_GLOBAL
    udQuot  : UDINT;
    udRest  : UDINT;
END_VAR

FUNCTION VIRTUAL GLOBAL NewClass0::Init

    IF _firstscan THEN

        IF _LSL_POS^.piSSR^.SetHandler THEN
            OS_SSRI_SetHandler(HANDLERTYPE_EXCEPTION, #EvalException(), this$pVoid);
        END_IF;

    END_IF;

END_FUNCTION //VIRTUAL GLOBAL NewClass0::Init

FUNCTION VIRTUAL GLOBAL NewClass0::CyWork
VAR_INPUT
    EAX      : UDINT;
END_VAR
VAR_OUTPUT
    state   : UDINT;
END_VAR
VAR
    udA      : UDINT;
    udB      : UDINT;
END_VAR

    udA := 4;
    udB := 0;
    udQuot := udA / udB;
    udRest := udA MOD udB;

    state:= READY;

END_FUNCTION //VIRTUAL GLOBAL NewClass0::CyWork
```

```

FUNCTION __CDECL  NewClass0::EvalException
VAR_INPUT
    excptCode  : UDINT;
    excptInfo  : ^EXCEPTION_POINTERS;
    taskType   : UDINT;
    pThis      : pVoid;
    param      : pVoid;
END_VAR
VAR_OUTPUT
    retVal     : DINT;
END_VAR

IF (_LSL_POS^.piSSR^.SkipOverIDIV)
    & (excptCode = EXCEPTION_INT_DIVIDE_BY_ZERO) THEN
    retVal := OS_SSR_SkipOverIDIV(excptCode,
                                    excptInfo$pVoid,
                                    16#99999999, // quotient
                                    16#88888888 // rest
                                );
ELSE
    retVal := EXCEPTION_OSHANDLER;
END_IF;

END_FUNCTION //__CDECL  NewClass0::EvalException

```

1.4 Command Line Interface CLI

1.4.1 Übersicht

1.4.1.1 Dateisystembefehle

<u>ARCH</u>	<u>ATTRIB / ATTR</u>	<u>CD / CHDIR</u>	<u>CHKDSK</u>
<u>COPY / C</u>	<u>CREATE</u>	<u>DEL / ERASE</u>	<u>DIR / D</u>
<u>DRIVES</u>	<u>FC</u>	<u>FORMAT</u>	<u>LABEL</u>
<u>MD / MKDIR</u>	<u>MOUNT</u>	<u>PARTITION</u>	<u>RD / RMDIR</u>
<u>REN</u>	<u>SMBSVR</u>	<u>TREE</u>	<u>TYPE</u>
<u>UMOUNT</u>	<u>UNZIP</u>	<u>VOL</u>	<u>XCOPY</u>
<u>ZIP</u>			

1.4.1.2 Betriebssystembefehle

ADDR	APPTASKS	ARP	BGTASKS
BIOS	BOOT	BOXES	BUFFERS / BUF / B
CANINFO	CIL	CYTASKS	DATE
DUMP	ERRORLOG	EXCEPT	FCLOSEALL
FILE	FILES	FLUSHLOG	GET ERRORLOG
HANDLES	HEAP	HELP / ?	INFO
INTS	LINK	LISTTS	LOG
MEM	MEMDUMP	NETSTAT	NETWORK
NUMLOCK	PACKAGE	PING	PNPBIOS
QUCMD	REBOOT	REXX	ROUTE
RTTASKS	SEMAS	SET CYCLIC	SET DBGHEAP
SET DEBUG	SET ERRORLOG	SET EVENTLOG	SET SYSTRACE
SET TRACEBUFSIZE	SET USB CACHE FLUSHING	SETENV	SETLEDSTATUS
SHOWENV	SRAMLOAD	SRAMSAVE	STATUS
TASKS1 / TASKS	TASKS2	VER	VT

1.4.1.3 LASAL data file (active.dat) Befehle

FILE	LOG		
----------------------	---------------------	--	--

1.4.1.4 Kernel Error Log Befehle

ERRORLOG	FLUSHLOG	GET ERRORLOG	SET ERRORLOG
--------------------------	--------------------------	------------------------------	------------------------------

1.4.1.5 Projektbefehle

SET FPUROUNDCONTROL	LSLOAD / LO	LSLSAVE / SA	PROJECT
RESET	RUN		

1.4.1.6 Konfigurationsbefehle

<u>ADDDBR</u>	<u>EURO</u>	<u>IPCINFO</u>	<u>SET</u>
<u>SET APPCODE</u>	<u>SET APPDATA</u>	<u>SET CLI</u>	<u>SET DATE</u>
<u>SET DBGLEVEL</u>	<u>SET DISPLAYROTATE XX</u>	<u>SET DISPLAYRESOLUTION</u>	<u>SET IP PORT</u>
<u>SET KEYS</u>	<u>SET LCD</u>	<u>SET MAINTIMER</u>	<u>SET OSHEAP</u>
<u>SET OSZIMEM</u>	<u>SET RAM</u>	<u>SET RUNTIME</u>	<u>SET RTRUNTIME</u>
<u>SET BTRUNTIME</u>	<u>SET ROUNTINGTABLE</u>	<u>SET SRAMRETAIN</u>	<u>SET STATION</u>
<u>SET STOPWAIT TIME</u>	<u>SET TIME</u>	<u>SET VISU</u>	<u>SET WATCHDOG</u>
<u>SET APPLSAVE</u>	<u>SET FORCEXTSINGLE</u>	<u>SCREENSAVER</u>	<u>US</u>

1.4.1.7 Konfigurationsbefehle (IP)

<u>IP</u>	<u>SET IP</u>		
-----------	---------------	--	--

1.4.1.8 Konfigurationsbefehle (CAN)

<u>SET CAN</u>			
----------------	--	--	--

1.4.1.9 Konfigurationsbefehle (DIAS)

<u>DIAS</u>	<u>SET DIASERROR</u>		
-------------	----------------------	--	--

1.4.1.10 Konfigurationsbefehle (SERIAL)

<u>SET LASALCOM</u>			
---------------------	--	--	--

1.4.1.11 Konfigurationsbefehle (Maus/Touch)

<u>CALIB</u>	<u>SET MOUSE</u>	<u>TOUCHCFG</u>	<u>TOUCHTEST</u>
<u>UPDATETOUCH</u>			

1.4.1.12 Konfigurationsbefehle (Drucker)

SET PRINTER			
-----------------------------	--	--	--

1.4.1.13 Batch-Verarbeitungsbefehle

CLS	DELAY	ECHO / @ECHO	EXIT
GOTO	IF EXISTS	PAUSE	REM

1.4.1.14 Umgebungsvariablen

FILENAMES	REBOOTONPF		
---------------------------	----------------------------	--	--

1.4.1.15 Command Line Syntax

Ein CLI (Command Line Interface) Befehl, einschließlich der erforderlichen bzw. optionalen Parameter und Schalter, muss in dem folgenden Standard-Format geschrieben werden:

BEFEHL erforderliche Parameter [optionale Parameter] [/Schalter]

Alle Artikel, die in eckigen Klammern stehen sind optional.

1.4.1.16 Tasten und Dateien

F2	Wiederholt die letzte Taste an dieser Stelle
F3	Wiederholt den letzten Befehl von dieser Position
ESC	Löscht die Eingabezeile
BSPC	Löscht das letzte Eingabezeichen
STRG-ALT-F1	Stellt US-Tastatur-Mapping ein
STRG-ALT-F2	Stellt deutsches Tastatur-Mapping ein
cmd> filename	Leitet die Befehlausgabe an eine Datei weiter
File.bat	Führt die Batch-Datei aus
AUTOEXEC.LS	Startup Batch-Datei (Sucht in A: \, dann C: \)
L	

? Zeigt alle CLI-Befehle

1.4.1.17 Reihenfolge der Ausführung von Befehlen in autoexec.lsl

Wenn LasalOS eine autoexec.lsl-Datei während des Boot-ups ausführt, werden nicht alle Befehle in der Reihenfolge durchgeführt, in der sie in der Datei erscheinen. Mehrere Befehle werden intern in der Warteschlange geordnet und dann nach der Verarbeitung der autoexec.lsl ausgeführt. Die Befehle, die in der Warteschlange stehen und die, die nicht in der Warteschlange sind, werden in der Reihenfolge durchgeführt, wie sie in der autoexec.lsl erscheinen.

Die Warteschlange enthält folgende Befehle:

[CHKDSK](#), [RESET](#), [BGTASKS](#), [CYTASKS](#), [RTTASKS](#), [APPTASKS](#), [TASKS](#), [TASKS1](#), [TASKS2](#), [PROJECT](#), [BOOT](#), [REBOOT](#), [LSLSAVE](#), [LSLOAD](#), [LINK](#), [CALIB](#) und [RUN](#).

Das folgende Beispiel erfordert Aufmerksamkeit.

```
BOOT D:x.rtb  
PAUSE
```

Hier wird versucht, ein neues Boot-Image zu installieren und anschließend die Ausführung zu unterbrechen. Der [BOOT](#)-Befehl wird der Warteschlange übertragen, so dass [PAUSE](#) vor [BOOT](#) ausgeführt wird. Dieser Prozess unterscheidet sich von dem, was die beiden Zeilen andeuten.

Um diesen Fehler zu vermeiden, wird einen zusätzlichen Befehl namens [QUCMD](#) aufgerufen, der den Befehl in der Warteschlange speichert:

```
BOOT D:x.rtb  
QUCMD PAUSE
```

Mit diesem Befehl wird [BOOT](#) automatisch in der Warteschlange gespeichert und [PAUSE](#) wird nach [BOOT](#) in der Warteschlange übertragen, so dass er nach der [BOOT](#)-Anweisung ausgeführt wird.

1.4.2 Dateisystembefehle

Die Befehle [CD/CHDIR](#), [COPY/C](#), [CREATE](#), [DEL/ERASE](#), [DIR/D](#), [FC](#), [MD/MKDIR](#), [RD/RMDIR](#), [REN](#), [TYPE](#) und [XCOPY](#) erfordern mindestens einen Parameter für den Pfad oder den Dateinamen. Ab LasalOS V01.01.049, kann der Pfad oder Dateinamen in Anführungszeichen ("") gestellt und Leerzeichen eingebettet, z. B.

```
DEL "c:\Pfad mit Leerzeichen\Dateiname mit Leerzeichen.txt"
```

Pfad- und Dateinamen, die mit Leerzeichen durch LasalOS erstellt wurden, werden immer als lange Dateinamen behandelt. Die kurzen Namen, die automatisch durch die Systeme

erstellt werden, enthalten keine Leerzeichen und sind etwas anders als die Dateinamen, die von anderen Betriebssystemen erstellt werden.

Die Umgebungsvariable [FILENAMES](#) kann auf LONG eingestellt werden, so dass die [DIR](#) Befehl den langen Dateinamen zeigen.

1.4.2.1 ARCH

Der Befehl dient zum Archivieren und Entpacken von Dateien und Ordnern.

Kurzform: ARCH

Syntax

```
ARCH <filename>
-r<root-dir> - Quellverzeichnis der zu archivierenden Datei
-f<file-spec> - Filename der zu archivierenden Datei (Wildcards werden unterstützt)
-x - Entpackt eine archivierte Datei
-s<split-size> - Spaltet das File in mehrere kleine Files auf
```

Beispiel 1

Archivierung

In folgendem Beispiel wird ein einzelnes Dokument archiviert und in einem anderen Zielordner gespeichert. Die zu archivierende Datei text.doc liegt im Ordner C:\DocFolder\ und das Archiv soll im Ordner C:\ZipFolder\ unter dem Namen TextZip.zip erstellt werden.

```
ARCH C:\ZipFolder\TextZip.zip -rC:\DocFolder -ftext.doc
```

Um eine gesamte Ordnerstruktur zu archivieren, wird der gleiche Befehl ohne Datei-Spezifikation angegeben.

```
ARCH C:\ZipFolder\TextZip.zip -rC:\DocFolder
```

Entpacken

Um das Archiv TextZip.zip im Ordner C:\ZipFolder in den Ordner C:\UnzipFolder zu entpacken, geben Sie folgenden Befehl ein.

```
ARCH C:\ZipFolder\TextZip.zip -rC:\UnzipFolder -x
```

Beispiel 2

Archivierung

In folgendem Beispiel wird eine gesamte Ordnerstruktur archiviert und in einem anderen Zielordner gespeichert. Die zu archivierenden Dateien liegen im Ordner C:\DocFolder\ und das Archiv soll im Ordner C:\ZipFolder\ unter dem Namen TextZip erstellt werden. Das Archiv soll mittels der Option -s in einzelne Teile mit der Größe 1 000 000 Bytes aufgeteilt werden. Hinweis: die Einzelteile des Archivs werden geringfügig größer als 1 000 000 Bytes, da jeder Teil mit einem Header versehen wird.

```
ARCH C:\ZipFolder\TextZip -rC:\DocFolder -s1000000
```

Entpacken

Um das geteilte Archiv TextZip im Ordner C:\ZipFolder in den Ordner C:\UnzipFolder zu entpacken, geben Sie folgenden Befehl ein.

```
ARCH C:\ZipFolder\TextZip -rC:\UnzipFolder -x -s1000000
```

Anforderungen

Der Befehl benötigt die DLM lslzip.dlm.

1.4.2.2 ATTRIB/ATTR

Der Attrib Befehl wird zum Anzeigen, Einstellen oder Entfernen eines oder mehrerer der vier Attribute (read-only, Archiv, System und versteckte) verwendet, die in Dateien und Verzeichnissen zugeordnet werden können. Normalerweise wird er verwendet, um Read-only-, versteckte und Systemattribute zu entfernen, sodass eine Datei verschoben, gelöscht oder eingestellt werden kann, oder eben nicht.

Kurzform: ATTR

Syntax

Um die Attribute eines Verzeichnisses anzuzeigen.

```
ATTRIB Verzeichnisname
```

Um die Attribute einer Datei anzuzeigen.

```
ATTRIB Dateiname
```

Um die Attribute einer Datei oder eines Verzeichnisses einzustellen oder zu entfernen

```
ATTRIB [+|-R] [+|-A] [+|-S] [+|-H] directory|filename
```

+	setzt ein Attribut
-	löscht ein Attribut
A	Archiv Dateiattribut
H	Verstecktes Dateiattribut
R	Read-only Dateiattribut
S	System Dateiattribut

Beispiel

Um die Attribute einer Datei mit dem Namen NEWS86 anzuzeigen, die sich auf dem aktuellen Laufwerk befindet, geben Sie folgenden Befehl ein.

```
ATTRIB news86
```

Um die Read-Only-Attribut in der Datei REPORT.TXT anzugeben, geben Sie folgenden Befehl ein.

```
ATTRIB r report.txt
```

1.4.2.3 CD/CHDIR

Ändert das aktuelle Standard-Verzeichnis.

Kurzform: CD

Syntax

```
CHDIR [Drive:][Path]
```

Beispiel

Jeder der folgenden Befehle ändert das aktuelle Verzeichnis in das Verzeichnis mit dem Namen LSLWORK.

```
chdir \lslwork  
cd \lslwork
```

Angenommen, Sie haben ein Verzeichnis mit dem Namen MPC mit einem Unterverzeichnis mit dem Namen IMAGES. Um Ihr aktuelles Verzeichnis in \MPC\IMAGES zu ändern, geben Sie folgenden Befehl ein.

```
cd \mpc\images
```

Oder, wenn das aktuelle Verzeichnis \MPC ist, können Sie den folgenden Befehl verwenden, um in das \MPC\IMAGES Verzeichnis zu wechseln.

```
cd images
```

Um von einem Unterverzeichnis zurück zum Hauptverzeichnis zu wechseln, geben Sie folgenden Befehl ein.

```
cd..
```

Zum Wechsel ins Hauptverzeichnis, geben Sie den folgenden Befehl ein.

```
cd\
```

1.4.2.4 CHKD SK

Prüft das Dateisystem des angegebenen Laufwerks auf Fehler und korrigiert sie optional.

Syntax

```
CHKDSK drive : [ /F ]
```

Drive:- Der Laufwerk-Buchstaben des zu überprüfenden Laufwerks.

/F- Wenn angegeben, wird versucht die Fehler zu korrigieren

Beispiel

Der folgende Befehl kontrolliert Laufwerk C:

```
Chkdsk C:
```

Der folgende Befehl prüft Laufwerk D: und korrigiert die Fehler.

```
Chkdsk D: /F
```



Dieser Befehl wird verwendet, um das Dateisystem eines Antriebs für Fehler zu kontrollieren. Einige Fehler können mit der Option /F korrigiert werden. Manchmal kann es erforderlich werden, chkdsk mit der Option /F zweimal auszuführen um Fehler zu löschen. Enthält das Dateisystem noch Fehler, wird es nicht mehr zuverlässig und sollte neu formatiert und initialisiert werden.

1.4.2.5 COPY/C

Kopiert eine Datei an einen bestimmten Zielort ohne Überprüfung von vorhandenen Dateien.

Kurzform: C

Syntax

```
COPY source [Zeil]
```

source- Dateiname der Datei, die kopiert werden soll.

destination- Dateiname bzw. das Verzeichnis der neuen Datei, Standard ist "..".

Beispiel

Dieser Befehl kopiert eine Datei.

```
Copy memo.doc brief.doc
```

So kopieren Sie eine Datei mit dem Namen ROBIN.TYP aus dem aktuellen Laufwerk und Verzeichnis auf ein bestehendes Verzeichnis mit dem Namen VOGEL, das sich auf Laufwerk C: befindet.

```
copy robin.typ c:\vogel
```

Existiert das Verzeichnis VOGEL nicht, kopiert LASALOS die Datei ROBIN.TYP in eine Datei namens VOGEL, die sich auf der Festplatte in Laufwerk C: im Hauptverzeichnis befindet.

Bemerkungen

Das LasalOS versucht die Zielfile in benachbarten Sektoren der Festplatte zu allozieren, um die Festplatte-Fragmentation zu reduzieren und die Zugangsgeschwindigkeit der Datei zu verbessern. Wenn nicht möglich, wird die Textmeldung "Information: file not contiguous" angezeigt. Obwohl dies weder ein Fehler noch eine Warnung ist, ist es ein Hinweis darauf, dass die Festplatte entweder stark fragmentiert oder voll ist.

Eine weitere Meldung, die angezeigt werden könnte, ist "SRAM full". Dies ist eine echte Fehlermeldung, die bedeutet, dass die Zielfile nicht erstellt werden konnte und weist darauf hin, dass die Zielfestplatte neu formatiert werden muss.

Beide Themen sind im LasalOS Benutzerhandbuch, Kapitel "How to setup/defragment a Compact-Flash-Medium" beschrieben.

1.4.2.6 CREATE

Erstellt eine neue Datei mit einer definierten Größe, aber undefiniertem Inhalt.

Syntax

```
CREATE filename file-size allocated-size
filename- Dateiname der zu erstellenden Datei.
```

file-size - Aktuelle Länge der Datei.

allocated-size- Zugewiesene Länge der Datei, zusammenhängend.

Beispiel

Der folgende Befehl bezieht sich auf eine Datei.

```
Create c:\tmp.log 0 20000000
```

Die Datei hat eine Länge von 0 Byte und ist in einem neu zugewiesenen Platz von etwa 20 Mb.



- Der Inhalt der Datei ist immer undefiniert.
- Dieser Befehl wird zu Performance-Zwecken verwendet:
- Die Suchoperationen werden auf HD und FDD beschränkt.
- Auf der C-IPC mit Dateien im Committed-Modus, muss die FAT nicht nach der Änderung der eigentlichen Datei-Größe neu geschrieben werden, nur das Verzeichnis muss aktualisiert werden.
- Wenn die Größe der Datei die vorbelegte Größe überschreitet, wird sie auf eine standard, nicht zusammenhängende Art erweitert und die Leistung für die Suchoperationen und FAT-Updates sinkt.

1.4.2.7 DEL/ERASE

Löscht eine Datei.

Kurzform: DEL

Syntax

```
ERASE filename
```

filename - Name der zu löschen Datei.

1.4.2.8 DIR/D

Dieser Befehl gibt eine Liste der Dateien und Verzeichnisse aus.

Kurzform: D

Betriebssystem RTK

DIR ohne Parameter zeigt als Kopfzeile das Laufwerk und gegebenenfalls den Namen eines Verzeichnisses. Z.B. Der Befehl „dir mpc“ erzeugt die Kopfzeile „Directory of C:\mpc“.

Darunter wird pro Verzeichnis oder Datei eine Zeile mit den unten beschriebenen Informationen ausgegeben.

Für ein Verzeichnis:

Name plus Erweiterung, Kennung „<DIR>“, Datum und Uhrzeit als das Verzeichnis erstellt wurde, die Attribute.

Für eine Datei:

Name (8 Zeichen) plus Erweiterung (3 Zeichen), Dateigröße in Bytes, Datum und Uhrzeit der letzten Änderung des Dateiinhalts, die Attribute.

In der Fußzeile werden die Gesamtzahl der Dateien und deren Gesamtgröße in Bytes ausgegeben.

Syntax

DIR [Laufwerk:] [/W] [/X] [/P] [/<attr>+|-] [Pfad | Dateiname]

/W - Zeigt nur die Namen im Spaltenformat, mit bis zu fünf Dateinamen und Verzeichnisnamen in jeder Zeile. Verzeichnisnamen stehen in eckigen Klammern.

/X - Erweiterte Anzeige. In der Kopfzeile wird zusätzlich das Volume-Label ausgegeben. Für Dateien wird zusätzlich der benötigte Platz auf dem Datenträger in Bytes ausgegeben sowie die erste Cluster-Nummer der Datei und die Anzahl der Cluster-Ketten. In den Fußzeilen wird zusätzlich die Anzahl der freien Bytes auf dem Datenträger ausgegeben.

/P - Pause nach jeder vollen Bildschirmseite.

/<attr>+|- Include/exclude Attribut. Gültige Werte für <attr> sind: H, S, R, A, V, D.

path | file name - die anzuzeigenden Pfade und Dateinamen können die Wildcard-Zeichen (*, ?) enthalten.

Betriebssystem Salamander

DIR ohne Parameter zeigt als Kopfzeile das Laufwerk und gegebenenfalls den Namen eines Verzeichnisses. Z.B. der Befehl „dir mpc“ erzeugt die Kopfzeile „Directory of C:\mpc“.

Darunter wird pro Verzeichnis oder Datei eine Zeile mit den unten beschriebenen Informationen ausgegeben.

Für ein Verzeichnis:

Datum und Uhrzeit, als das Verzeichnis erstellt wurde, Kennung „<DIR>“, Verzeichnisname plus Erweiterung.

Für eine Datei:

Datum und Uhrzeit der letzten Änderung des Dateiinhalts, Dateigröße in Bytes, Dateiname plus Erweiterung.

In den Fußzeilen werden die Gesamtzahl der Dateien und deren Gesamtgröße in Bytes sowie die Anzahl der Verzeichnisse ausgegeben.

Syntax

DIR [Laufwerk:] [/W] [/S] [/X] [/P] [Pfad | Dateiname]

/W - Zeigt nur die Namen im Spaltenformat, mit bis zu fünf Dateinamen und Verzeichnisnamen in jeder Zeile. Verzeichnisnamen stehen in eckigen Klammern.

/S - Die Ausgabe entspricht dem Format von DIR ohne Optionen im Rtk-Betriebssystem.

/X Erweiterte Anzeige. In den Fußzeilen werden zusätzlich ausgegeben:

Total: - Datenträgergröße in Bytes - Anzahl der Cluster auf dem Datenträger

Free: - Anzahl der freien Bytes - Anzahl der freien Cluster

SectorsPerClusters: -Anzahl der Sektoren pro Cluster

BytesPerSector: - Anzahl der Bytes pro Sektor

/P - Pause nach jeder vollen Bildschirmseite. Diese Option ist bei Remote-CLI nicht wirksam.

path | file name - die anzuzeigenden Pfade und Dateinamen können die Wildcard-Zeichen (*, ?) enthalten.

1.4.2.9 DRIVES

Zeigt eine Liste der verfügbaren Laufwerke.

Syntax

DRIVES

- versucht, alle Laufwerke zu installieren und listet alle Laufwerke, die erfolgreich installiert wurde. Floppy-Laufwerke ohne eine eingefügte Festplatte werden nicht gelistet, denn sie können nicht installiert werden.

DRIVES RAW

- Listet alle möglichen Laufwerke, auch wenn sie nicht physisch anwesend sind und zeigt ihren aktuellen Zustand, ohne ihn zu verändern.

DRIVES PHYSICAL

- versucht alle Laufwerke zu installieren und listet alle Laufwerke, einschließlich Disketten, die physisch vorhanden sind.

Anforderungen

LasalOS: ab LasalOS 5,50 ist erforderlich.

1.4.2.10 FC

Vergleicht den Inhalt von zwei Dateien und zeigt die Unterschiede zwischen den beiden an.

Syntax

FC filename1 filename2

filename1- Name den zu vergleichenden Dateien.

filename2- Name der mit dateiname1 zu vergleichender Datei oder das Verzeichnis zum Suchen einer Datei mit dem gleichen Namen wie dateiname1.

1.4.2.11 FORMAT

Formatiert einen logischen Drive.

Syntax

FORMAT Drive

Beispiel

format a:

Der FORMAT-Befehl erstellt ein neues Hauptverzeichnis und eine neue File Allocation Table für die Festplatte. Er kann auch auf schlechte Bereiche auf der Festplatte prüfen sowie alle Daten von der Festplatte löschen. Um eine neue Festplatte mit LasalOS verwenden zu können, müssen Sie zunächst diesen Befehl zum Formatieren der Festplatte verwenden.

1.4.2.12 LABEL

Erstellt, ändert oder löscht ein Volume Label einer Festplatte.

Syntax

LABEL drive: label

Ist kein Label angegeben, werden alle bestehenden Label entfernt.

1.4.2.13 MD/MKDIR

Erstellt ein neues Verzeichnis.

Kurzform: MD

Syntax

MKDIR pathname

pathname - Name des zu erstellenden Verzeichnisses.

Beispiel

Angenommen, Sie möchten ein Verzeichnis auf der Festplatte im aktuellen Drive erstellen. Um ein Verzeichnis mit dem Namen Mydir zu erstellen, geben Sie folgenden Befehl ein.

mkdir \mydir

Sie können diesen Befehl mit den gleichen Ergebnissen auch folgendermaßen eingeben.

md mydir

Jetzt nehmen wir an, dass das Mydir-Verzeichnis das aktuelle Verzeichnis ist, und dass Sie ein Unterverzeichnis von Mydir namens PROJEKTE erstellen möchten. Um das PROJEKTE Verzeichnis zu erstellen, geben Sie folgenden Befehl ein.

```
mkdir Projekte
```

1.4.2.14 MOUNT

Installiert eine Windows- oder eine Linux Samba Freigabe als Laufwerk. Es können auch lokale Verzeichnisse als Laufwerke installiert werden. Dieses Kommando ist nur unter Salamander verfügbar.

Syntax

```
MOUNT type netpath lokales Laufwerk -uuser -ppasswort -RO -SYNC -ttimeout
```

type – Typ des Mounts. SAMBA, SAMBA_V1, SAMBA_V2 or LOCAL. Mit den Typen SAMBA_V1 und SAMBA_V2 wird die jeweilige Samba Protokollversion eingestellt. Wird als *<type>* SAMBA angegeben, dann wird Samba-Version 1 verwendet. Es kann hier auch samba_string angegeben werden. Dann können diverse Einstellungen (Samba Protokollversion, Username, Password etc.) mit einem Linux Option String vorgenommen werden. Siehe -o.

netpath – Netzwerkpfad der Freigabe. (IP-Adresse, DNS-Name oder NETBIOS Name \Freigabe) oder lokales Verzeichnis.

lokales Laufwerk – Das Laufwerk, wo die Daten des Netzlaufwerks zur Verfügung gestellt werden

-u – Der Username des Mountpfad. (optional).

-p – Das Passwort des Mountpfad. (optional).

-o – Linux option string (-o und -u, -p schließen einander aus). Hier können eine ganze Reihe von Linux-Optionen für Samba angegeben werden (z.B. sec, vers, username, password, domain etc.). Der String darf keine Leerzeichen enthalten. Die einzelnen Optionen müssen mit einem Beistrich getrennt werden. Siehe Beispiel. -o kann nur zusammen mit *<type>* = samba_string verwendet werden.

-RO – Das Netzlaufwerk wird readonly installiert. (optional und nur für Netzlaufwerke)

-SYNC – Schreibzugriffe werden nicht gepuffert, sondern sofort ausgeführt. (optional und nur für Netzlaufwerke)

-t – Timeoutzeit in ms für die Auflösung von NETBIOS-Namen. Die Timeoutzeit wird nur verwendet um NETBIOS-Namen aufzulösen. Das Installieren selbst wird nicht in diese Zeit miteingerechnet. (optional, default = 2 s).

Beispiel

Mit folgendem Befehl wird die Freigabe DSWWXX/MountTest als Laufwerk D: installiert. Die Timeout Zeit für die Namesauflösung wird auf 2 s gestellt.

```
mount samba \\DSWWZZ\MountTest d: -uMountUser -pMountUser -t2000
```

Mit folgendem Befehl wird die Freigabe DSWWZZ/MountTest als Laufwerk E: installiert. Es wird die Samba Protokollversion 2.0 verwendet. Die Timeout Zeit für die Namesauflösung wird auf 2 s gestellt. Schreibzugriffe werden nicht gepuffert.

```
mount samba_string \\DSWWZZ\MountTest E: -overs=2.0,username=MountUser,password=MountUser -SYN
```

1.4.2.15 PARTITION

Erstellt eine einzige Partition über die gesamte Disk. Die Partition wird aktiviert. Alle Daten können zerstört werden.

Syntax

```
PARTITION pdi
```

pdi – Physischer Disk-Index, dieser Wert zur gewünschten Festplatte kann mit dem Befehl "DRIVES PHYSICAL" in der Spalte "PDI" der Ergebnisliste gefunden werden. Gültige Werte liegen im Bereich von 0 bis 25.

Anforderungen

LasalOS: ab LasalOS 5.50 ist erforderlich.

1.4.2.16 RD/RMDIR

Entfernt ein Verzeichnis.

Kurzform: RD

Syntax

```
RMDIR pathname
```

Pfadname - Name des zu entfernenden Verzeichnisses

Beispiel

Um ein Verzeichnis mit dem Namen \Benutzer\SMITH zu löschen, stellen Sie zunächst sicher, dass der Ordner wie im folgenden Beispiel leer ist.

```
dir \user\smith
```

LASALOS sollte nur die "." und ".." Symbole zeigen.

Dann, aus einem beliebigen Verzeichnis außer \Benutzer\SMITH, geben Sie folgenden Befehl ein.

```
rmmdir \user\smith
```

Sie können den folgenden Befehl mit dem gleichen Ergebnis auch folgendermaßen eingeben.

```
rd \user\smith
```

1.4.2.17 REN

Ändert den Namen der Datei oder Dateien, die Sie angeben.

Syntax

```
REN filename newname
```

filename - Name der zu umbenennenden Datei oder Verzeichnis.

newname - Neuer Name für die angegebene Datei oder Verzeichnis.

Beispiel

Um eine Datei mit dem Namen CHAP10 auf Part10 umzubenennen, geben Sie folgenden Befehl ein.

```
ren b:chap10 part10
```

Die neu umbenannte Datei Part10 bleibt auf Laufwerk B.

1.4.2.18 SMBSVR

Mit dem Packet SAMBA-Server kann anderen Steuerungen der Zugriff auf das eigene Dateisystem gewährt werden. Siehe auch den Befehl MOUNT. Mit dem Befehl SMBSVR werden die Benutzer und die Dateifreigaben (Shares) verwaltet.

Syntax

```
SMBSVR START
```

startet den SMB-Server

```
SMBSVR STOP
```

stoppt den SMB-Server

```
SMBSVR USER ADD username password r/w path
```

richtet einen SMB-User ein
usernameName des SMB-Users
passwordPassword des SMB-Users
r/wgenereller Schreib- oder Lesezugriff
pathHeimatverzeichnis des SMB-Users



Bei älteren Salamander-Versionen (\leq 09.03.120) wird das User-Verzeichnis beim Löschen des Samba-Users ebenfalls gelöscht. Man sollte daher beim Parameter <path> nicht C:\ als User-Verzeichnis angeben!

SMBSVR USER REMOVE username
löscht einen SMB-User

username - Name des SMB-Users

SMBSVR ACCESS ADD username/any sharename path r/w
richtet eine SMB-Freigabe (Share) ein

username - Name des SMB-Users; Wird any angegeben, kann man sich mit einem beliebigen, vorherdefinierten Benutzer verbinden.

sharenameName der Freigabe (Share)

path - Verzeichnis, das freigeben werden soll

r/wLese- oder Schreibzugriff

Das Einrichten der Freigabe kann bis zu 90 s dauern. Soll es sofort ausgeführt werden, muss man den SBM-Server stoppen und wieder starten.

SMBSVR ACCESS REMOVE sharename
löscht eine SMB-Freigabe (Share)

sharenameName der Freigabe (Share)

Das Löschen der Freigabe kann bis zu 90 s dauern. Soll es sofort ausgeführt werden, muss man den SBM-Server stoppen und wieder starten.

Beispiel

```
SMBSVR START
SMBSVR USER ADD testa testa w c:\testa
SMBSVR ACCESS ADD testa test-share-a c:\testa w
SMBSVR ACCESS ADD any test-share-b c:\ w
```

1.4.2.19 TREE

Zeigt eine komplett Verzeichnisbaum-Hierarchie.

Syntax

`TREE pathname`

pathname - Name des Verzeichnisses, mit dem die Anzeige gestartet wird.

1.4.2.20 TYPE

Zeigt den Inhalt einer Textdatei. Verwenden Sie den TYPE-Befehl, um eine Textdatei anzuschauen, ohne sie zu ändern.

Syntax

`TYPE [drive:][path]filename`

filename - Name der Datei, die angezeigt werden soll

Beispiel

Wenn Sie den Inhalt einer Datei mit dem Namen HOLIDAY.MAR zeigen möchten, geben Sie folgenden Befehl ein.

`type holiday.mar`

Die Anzeige einer Datei, die keine Textdatei ist (z.B. *.bin), wird nicht empfohlen und kann zu unerwartetem Verhalten führen.



1.4.2.21 UMOUNT

UMount eines gemounteten Netzlaufwerks oder lokalen Verzeichnisses. Dieses Kommando ist nur unter Salamander verfügbar.

Syntax

`UMOUNT lokales Laufwerk`

lokales Laufwerk – Das Laufwerk, wo die Daten des Netzlaufwerks zur Verfügung gestellt werden

Beispiel

Mit folgendem Befehl wird die auf Laufwerk D: bereitgestellte Freigabe getrennt.

umount d:

1.4.2.22 UNZIP

Der Befehl dient zum Entpacken (Entzippen) einer Zip-Datei.

Kurzform: UNZIP

Der Befehl benötigt die DLM lslzip.dlm.



Syntax

```
UNZIP <in-file> ... Dateiname/-pfad der Zip-Datei  
<dest-dir> ... Dateiname/-pfad der Zielfile des Entpackens
```

Beispiel

Die Zip-Datei zipFile.zip aus dem Ordner C:\ZipFolder soll als Datei dest.txt entpackt werden.

1.4.2.23 VOL

Zeigt Informationen über ein Laufwerk.

Syntax

```
VOL [pathname:]  
pathname - Name eines Laufwerks, dessen Informationen gezeigt werden sollen.  
Nur der Laufwerksbuchstabe wird vom Pfad genommen.
```

1.4.2.24 XCOPY

Kopiert Dateien und unterstützt Wildcards.

Syntax

`XCOPY source destination`

source - zu kopierenden Dateien (*. ext, *.*)

destination - Dateiname bzw. das Verzeichnis der neuen Dateien

Bemerkungen

Das LasalOS versucht die Zielfile in benachbarten Sektoren der Festplatte zu allozieren, um die Festplatte-Fragmentation zu reduzieren und die Zugangsgeschwindigkeit der Datei zu verbessern. Wenn nicht möglich, wird die Textmeldung "Information: file not contiguous" angezeigt. Obwohl dies weder ein Fehler noch eine Warnung ist, ist es ein Hinweis darauf, dass die Festplatte entweder stark fragmentiert oder voll ist.

Dies ist eine echte Fehlermeldung, die bedeutet, dass die Zielfile nicht erstellt werden konnte und weist darauf hin, dass die Zielfestplatte neu formatiert werden muss.

Beide Themen sind in der LasalOS Benutzerhandbuch, Kapitel "How to setup/defragment a Compact-Flash-Medium" beschrieben.

1.4.2.25 ZIP

Der Befehl dient zum Komprimieren (Packen/Zippen) von Dateien. Es kann immer nur eine Datei komprimiert werden. Wird eine vorhandene Zip-Datei als Ziel angegeben, wird die vorhandene durch die neue überschrieben.

Der Befehl benötigt die DLM `lslzip.dlm`.



Syntax

`ZIP <in-file> ... Dateiname/-pfad der zu archivierenden Datei`
`<out-file> ... Dateiname/-pfad der zu erstellenden Zip-Datei`
`<pack-level> ... Komprimierungslevel (1 = schnellste, ..., 9 = größte Kompression)`

Beispiel

Die Datei source.txt aus dem Ordner „C:\SourceFolder“ wird zu einer Zip-Datei mit Namen „zipFile.zip“ im aktuellen Ordner komprimiert.

```
ZIP C:\SourceFolder\source.txt zipFile.zip 6
```

1.4.3 Betriebssystembefehle

1.4.3.1 ADDR

Zeigt die Adresse einer verbundenen Variablen.

Syntax

```
ADDR Variable
```

Beispiel

```
addr OPS
```

Zum Anzeigen der Adresse der Variablen OPS.

1.4.3.2 APPTASKS

Zeigt alle Real-Time, Zyklischen und Background-Tasks der Applikation an.

Syntax

```
APPTASKS
```

1.4.3.3 ARP

Zeigt die physikalischen MAC-Adressen der angeschlossenen TCP-Clients.

Syntax

```
ARP
```

1.4.3.4 BGTASKS

Zeigt alle Background-Tasks der Applikation.

Syntax

BGTASKS

1.4.3.5 BIOS

Zeigt BIOS-Informationen an.

Syntax

BIOS

1.4.3.6 BOOT

Installiert ein neues Betriebssystem auf der Festplatte. Der folgende Hinweis gilt nur für Salamander-Betriebssysteme. Wenn die Versionsnummer des aktuell installierten Salamander-Betriebssystems im Bereich von 09.03.102 bis 09.03.143 liegt, dann kann es zu Problemen mit installierten SW-Paketen kommen. Es gibt zwei Möglichkeiten, um diese Probleme zu verhindern:

Option 1 Es müssen alle installierten SW-Pakete mit „PACKAGE UNINSTALL“ deinstalliert werden, bevor ein neues Salamander-Betriebssystem installiert wird. Nachdem das neue Salamander-Betriebssystem installiert wurde, müssen die SW-Pakete mit „PACKAGE INSTALL“ neu installiert werden. Details siehe unter dem CLI-Befehl [PACKAGE](#).

Option 2 Das neue Betriebssystem muss zweimal unmittelbar hintereinander installiert werden.

Syntax

BOOT [drive:]image [dest-drive:]

image - Dateiname des zu installierenden .LBI- oder .RTB-Images

dest-drive - Laufwerk, auf dem das neue Image installiert wird. Wenn nicht angegeben, wird das Abbild auf Laufwerk C: installiert

Bemerkungen

Ab Betriebssystemversion 01.01.096 kann ein neues Boot-Image Dateiformat verwendet werden. Das LBI (LASAL Boot-Image)-Format enthält ein Betriebssystem-Image und

Sicherheitsmechanismen, wie z. B. eine ID für die Ziel-Plattform und eine CRC32, um die Installation eines fehlerhaften OS-Images zu vermeiden.



Ab LasalOS 01.02.096 sollten nur noch LBI-Dateien verwendet werden!

Ab LasalOS 01.02.150 werden nur noch LBI-Dateien unterstützt!

Beispiel

```
BOOT C:\etvedge.lbi
```

Rückgabewerte

Die folgenden Fehler-Codes können zurückgegeben werden, wenn der Befehl über ein Rexx-Skript verwendet wird:

Wert	Beschreibung
-1	Interner Fehler, kann nicht SPS-Typ feststellen
-2	Ungültiger Parameter (kein Dateiname angegeben, Ungültiger Drive, ...)
-3	Datei nicht gefunden
-4	LBI nicht ausreichend Arbeitsspeicher
-5	LBI falschen CRC
-6	LBI Ungültiger Header
-7	LBI-Datei ungültige Länge
-8	LBI ungültige Version
-9	LBI enthält ein ungültiges Image für die CPU
-10	LBI keine RTB-Image-Datei
-11	LBI Fehler beim Öffnen der Datei
-12	LBI-Fehler Datei schreiben
-13	Nicht ausreichend Speicher
-14	Interner Fehler, erstellen vollständiger Dateiname
-1001	Fehler beim Installieren eines neuen Images, nicht genügend Arbeitsspeicher
-1002	Fehler beim Installieren eines neuen Images, ungültig RTB-Datei
-1003	Fehler beim Installieren eines neuen Images, Gerätefehler

-1004	Fehler beim Installieren eines neuen Images, kein gültiges Boot-Code
-1005	Fehler beim Installieren eines neuen Images, das Boot-Image Datei, können auch zurückgegeben werden, wenn die Disk voll ist
-1006	Fehler beim Installieren eines neuen Images, nicht zusammenhängend
-1007	Fehler beim Installieren eines neuen Images Sektorgröße
-1008	Fehler beim Installieren eines neuen Images, Boot-Sektor schreiben
-1009	Fehler beim Installieren eines neuen Images, ungültig RTB-Datei
-1010	Fehler beim Installieren eines neuen Images, Boot-Code fehlerhaft
-1011	Fehler beim Installieren eines neuen Images, Fehler beim Öffnen der RTB-Datei
-1012	Fehler beim Installieren eines neuen Images, nicht unterstützt
-1013	Fehler beim Installieren eines neuen Images, nicht unterstützt
-1014	Fehler beim Installieren eines neuen Images, Puffer zu klein
-1015	Fehler beim Installieren eines neuen Images, Festplatte voll, -1005 können auch zurückgegeben werden, abhängig von der, Fragmentierung

Anforderungen

LasalOS: der Parameter "dest-drive" erfordert LasalOS ab 5.50. Frühere Versionen installieren das neue Image immer auf Laufwerk C:.

1.4.3.7 BOOTPIC

Einstellungen, wie das Boot-Image angezeigt werden soll

Syntax

BOOTPIC (Option) (Option) ...

Anzeige des Boot-Images aktivieren

Option	Beschreibung
SHOWCLI	deaktiviert "Press TAB to show the CLI"
AUTOEXEC	deaktiviert "Autoexec.ls1 finished"
QUITONCLISTART	deaktiviert das Boot-Image, nachdem CLI gestartet ist
NOT	invertiert alle gesetzten Optionen

Beispiel

```
BOOTPIC NOT QUITONCLISTART
```

Das Boot-Image ist aktiviert, alles ist ausgeschaltet und das Boot-Image muss manuell deaktiviert werden

1.4.3.8 BOXES

Zeigen Informationen über alle Mailboxen an.

Syntax

```
BOXES
```

1.4.3.9 BUFFERS/BUF/B

Zeigt Informationen über die Puffer-Cache der Dateien an.

Kurze Formen: BUF und B

Syntax

```
BUFFERS
```

1.4.3.10 CANINFO

Zeigt Informationen über die CAN-Einstellungen an.

Syntax

```
CANINFO ifnum
```

ifnum – nummer der CAN-Schnittstelle (1 = CAN1, 2 = CAN2, ...).

1.4.3.11 CIL

Zeigt alle registrierten Common-Interface-Libraries an.

Syntax

```
CIL
```

1.4.3.12 CYTASKS

Zeigt alle zyklischen Tasks der Applikation an.

Syntax

CYTAKTS

1.4.3.13 DATE

Zeigt das aktuelle Datum und die Uhrzeit an.

Syntax

DATE

1.4.3.14 DUMP

Dump Applikationsspeicher.

Syntax

DUMP Start [End]

Start - Start Adresse

End - End Adresse

Wenn die Endadresse nicht angegeben ist, wird ein Standardwert von Start + 15 angenommen. Die Adresswerte müssen in einem Hexadezimal-Format angegeben werden.

Es muss ein gültiger Arbeitsbereich vorhanden sein und der Adressbereich muss im User-Code- oder User-Data-Abschnitt des RAM liegen.

Beispiel

DUMP f002ec

Dump von 16 Byte ab Adresse F002ECh

DUMP f002ec f002ef

Dump von der Adresse F002ECh bis Adresse F002EFh

1.4.3.15 ERRORLOG

Eine Verknüpfung um die Kernel Fehler-Protokolle zu untersuchen.

Dieser Befehl wird die Protokolle in eine Datei TEMPERR.LOG speichern und sie dann seitenweise anzeigen. Danach wird der Temp-Datei gelöscht.

Syntax

ERRORLOG

1.4.3.16 EXCEPT

Zeigt Exception-Informationen an oder setzt sie zurück.

Syntax

EXCEPT

Zeigt Exception-Informationen an.

EXCEPT RESET

Setzt Exception-Informationen zurück.

1.4.3.17 FCLOSEALL

Schreibt alle gecacheten Puffer auf die Festplatte und schließt alle geöffneten Dateien.

Syntax

FCLOSEALL

1.4.3.18 FILE

LASAL Datendatei Befehle (active.dat).

Syntax

FILE LOAD

Lädt eine Datei herunter.

FILE LOAD ACTIVE

Lädt die Default Active Datendatei herunter.

FILE LOAD ACTIVE filename

Lädt die angegebene Datei als die ACTIVE Datei herunter.

FILE LOAD mask filename

Lädt die angegebene Datei mit der angegebenen HEX Maske.

FILE SAVE

Speichert eine Datei.

FILE SAVE ACTIVE

Speichert die default ACTIVE Datei.

FILE SAVE ACTIVE filename

Speichert die angegebene Datei als die ACTIVE Datei.

FILE SAVE mask filename

Speichert die angegebene Datei mit der angegebenen HEX Maske.

FILE DISPLAY

Zeigt eine Datei.

FILE DISPLAY filename

Zeigt die angegebene Datei.

1.4.3.19 FILES

Zeigt Informationen über alle geöffneten Dateien.

Syntax

FILES

Beispiel

FILES	Index	Handle	Flags	FilePos	Name
-----	3	00070003	W_S	1234	C:\ADIR\SOMEFILE.DAT
	4	00030004	RD_	256	C:\ADIR

Die Index Spalte enthält den Dateibereich, in dem die Datei sich in der internen File Table befindet; Handle ist die hexadezimale Darstellung des Datei-Handle. Die Flag Spalte kann eine Kombination der Buchstaben R, W, D und S zeigen, die nur Lesezugriff, Lese- und Schreibzugriff, Verzeichnis, sowie den gemeinsamen Zugang darstellen. FilePos gibt den aktuellen Wert des Dateizeigers an; Name ist der vollständige Pfad der Datei. Verwenden Sie diesen Befehl um zu analysieren, welche Dateien geöffnet sind, wenn ein Programm alle Dateien nicht richtig schließt.

1.4.3.20 FLUSHLOG

Spielt den Inhalt des Betriebssystempuffers in eine Datei.

Syntax

FLUSHLOG

1.4.3.21 GET ERRORLOG

Speichert den protokollierten Kernel-Fehler.

Dieser Befehl speichert den protokollierte Kernel-Fehler in eine angegebene Datei. Wenn kein Dateiname angegeben ist, wird ErrorLog.txt erstellt.

Syntax

GET ERRORLOG [file name]

1.4.3.22 HANDLES

Zeigt eine Liste aller derzeit vorhandenen Handles.



Diese Funktion steht nur unter den Betriebssystem RTK zur Verfügung.
Sie ist nur für internen Gebrauch!

Syntax

HANDLES

Beispiel

```
HANDLES
FreeHandles=106, FreeObjects=106, FreeTypes=21
1: cnt=1, name=
2: cnt=1, name=
3: cnt=1, name=Thread
4: cnt=1, name=Thread
5: cnt=1, name=Thread
6: cnt=1, name=Thread
7: cnt=1, name=Thread
8: cnt=1, name=Thread
9: cnt=1, name=Thread
10: cnt=1, name=Thread
```

```
11: cnt=1, name=Thread
12: cnt=1, name=Thread
13: cnt=1, name=Thread
14: cnt=1, name=Thread
15: cnt=1, name=Thread
16: cnt=1, name=Thread
17: cnt=1, name=Thread
18: cnt=1, name=Thread
19: cnt=1, name=Thread
20: cnt=1, name=Thread
21: cnt=1, name=
22: cnt=1, name=Mutex
```

1.4.3.23 HEAP

Zeigt Informationen über den Applikations-Heap.

Syntax

HEAP

Beispiel

```
HEAP
-Application Heap Info-
TotalSize: 41926356
TotalUsed: 69028
TotalFree: 41857328
```

1.4.3.24 HELP/?

Zeigt Hilfe-Informationen.

Kurzform: ?

Syntax

HELP

Zeigt alle CLI-Befehle.

HELP Befehl

Zeigt Informationen über den Befehl 'command'.

1.4.3.25 INFO

Zeigt Informationen zu bestimmten Themen. Dient vor allem zu diagnostischen Zwecken.

Syntax Betriebssystem RTK

INFO USB

Zeigt den USB-Host und angeschlossene Geräte an.

INFO INFOBLOCK

Zeigt den Sigmatek Device Info Block an.

INFO XREGS

Zeigt die Xilinx Register an.

INFO SIGMAIRQ

Zeigt die Sigmatek Device Interrupts an.

Syntax Betriebssystem Salamander

INFO SERNUM

Zeigt die Serien-Nummern diverser Geräte (z.B. CPU und SD-Karte) an.

INFO VER

Zeigt die Versionsnummern von Betriebssystem und Hardware (FPGA) an.

INFO MEM

Zeigt die momentane Speicherbelegung an.

INFO MEMREQUIRE

Zeigt den Speicherbedarf des Betriebssystems und der installierten SW-Pakete an.

INFO CONN

Zeigt Informationen über Schnittstellen und Verbindungen an

INFO SDCARD

Zeigt Informationen über die SD-Karte an

INFO CPUTEMP

Zeigt die CPU Temperatur(en) an

1.4.3.26 INTS

Zeigt Informationen über Interrupts seit dem Start des Programms.

Dieser Befehl steht nur beim Betriebssystem RTK zur Verfügung.



Syntax

INTS

Für jeden vom Betriebssystem registrierten IRQ, bei dem seit dem Programmstart Interrupts aufgetreten sind, wird eine einzelne Informationszeile ausgegeben. Angezeigt werden die IRQ-Nummer, die Anzahl der aufgetretenen Interrupts, der unbenutzte Interrupt-Stack und die Anzahl der rekursiven Interrupts (normalerweise 0). Die kumulierte CPU-Zeit für jeden IRQ wird nur in der Debug-Version des Betriebssystems zurückgegeben.

Wenn Rekursionen aufgetreten sind (wie beim Tastatur-Interrupt im Beispiel), wird auch eine Statistik für den Panic-Stack angegeben. Rekursionen auf dem Panic-Stack (Spalte "Doubles" in Zeile "Panic") bedeuten Gefahr und deuten auf einen Fehler im Interrupthandle oder eine Interrupt-Überlastung hin.

Beispiel

	IRQ	Calls	FreeStack	Doubles	Time
0	1194	188	0	0	0.103566
1	137	158	2	0	0.021934
3	6663	156	0	0	0.734534
Panic	2	158	0	0	0.004392

1.4.3.27 LINK

Zeigt die Linker-Informationen

Syntax

LINK INFO

Zeigt allgemeine Link-informationen.

LINK MODULES

Zeigt Informationen über alle verbundenen Module dieses Projekts

LINK GLOBALS

Zeigt Informationen über die globalen Symbole dieses Projekts

1.4.3.28 LISTTS

Zeigt Informationen über den Stack-Bereich der Betriebssystem-Tasks.

Syntax

LISTTS

Die erste Spalte zeigt den Namen des Tasks, dann zeigt eine mit einem Komma getrennte Liste die folgenden Werte an:

- Start- und Endadresse des Stacks
- den aktuellen Stack-Zeiger (ESP)
- die geringste Anzahl von freien Bytes auf dem Stack eines Tasks, seitdem er erstellt wurde

Beispiel

Main Task	:000D00000-00DBFF88,00DBFF84,783148d
Idle Task	:000B3E50-000B4050,000B4010,360d
Main Task Low	:000B4870-000B4D70,000B4D0C,1072d
Main Task High	:000C2190-000C3290,000C31D0,4144d
CPU Monitor	:000C4550-000C4750,000C46DC,304d
DLED_Task	:000C4B40-000C5040,000C4FEC,1072d
RT_AsyncTask	:000C6760-000C7860,000C77C8,4144d
UserLogTask	:000C8460-000CA560,000CA3A8,5716d
CanRxMailThread	:000CBF90-000CD090,000CCFA4,4116d
IPTASK	:03803480-03804580,038044C4,3712d
INTERRUPT	:03804AE0-03805BE0,03805B24,4120d
IPTASK	:03806140-03807240,03807184,4120d
INTERRUPT	:038077A0-038088A0,038087E4,4120d
IPTASK	:03808E00-03809F00,03809E44,4120d
INTERRUPT	:0380A460-0380B560,0380B4A4,4120d
TIMER	:0380BAC0-0380CBC0,0380CAF8,3680d
TCP Server	:0380D5F0-0380F6F0,0380F1B8,6504d
USB Hub	:03812430-03813530,03813450,2628d
USB Mouse	:03815640-03815F40,03815E80,2048d
USB Keyboard	:03816710-03817010,03816F50,1928d
DbgRxFast	:038C7050-038C9150,038C9020,8108d
WSP0_CT	:038D4FF0-038F50F0,038F5088,128268d
TCP Client 1	:0382AA10-0382EB10,0382DF74,12952d
CMD_LINE	:038174D0-0381D5D0,0381CED4,2544d

1.4.3.29 LOG

Aktiviert bzw. deaktiviert die Logging-Funktionen.

Syntax

```
LOG ACTIVE.DAT [value]  
LOG ACTIVE.DAT
```

Zeigt den Stand der ACTIVE.DAT Protokollierung.

```
LOG ACTIVE.DAT ON|OFF
```

Aktiviert/Deaktiviert das Schreiben auf die ACTIVE.LOG-Datei beim Schreiben der ACTIVE.DAT bei Reset/Power down.

1.4.3.30 MEM

Zeigt Informationen über die Heap-Speicher des Betriebssystems.

Syntax

```
MEM
```

1.4.3.31 MEMDUMP

Zeigt den Inhalt eines Speicherbereichs.

Syntax

```
MEMDUMP address length [filename]
```

Wird der Dateiname des optionalen Parameters angegeben, dann wird der Inhalt des Speichers in einem binären Format in eine Datei kopiert.

Beispiel

```
MEMDUMP 0x17c0000 64  
017C0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
017C0010: 00 00 00 00 00 00 00 00 00 00 E8 03 00 00 00 00  
017C0020: 78 47 7C 01 88 B8 F3 01 58 63 11 00 30 55 E2 01  
017C0030: 30 3F 81 03 00 00 00 00 00 00 00 00 75 47 00 00
```

1.4.3.32 NETSTAT

Zeigt Informationen über die aktuellen TCP-Verbindungen.

Syntax

NETSTAT

Beispiel

Prot	Local Host	Remote Host	Status
TCP	192.168.140.100:1954	192.168.140.140:1079	BUSY

Prot Protokoll der Verbindung

Local Host IP-Adresse der lokalen Maschine

Remote Host IP-Adresse des Remote-Clients

Status Kann READY, BUSY oder ERROR sein

1.4.3.33 NETWORK

Zeigt Informationen über installierte Ethernet-Geräte an.

Syntax

NETWORK

Zeigt Informationen über alle installierte Ethernet-Geräte an.

NETWORK iface

Zeigt Informationen über die Ethernet-Schnittstelle mit der Nummer iface

1.4.3.34 NUMLOCK

Zeigt oder ändert den aktuellen Status der Numlock-Tastatur.

Syntax

NUMLOCK

Zeigt die aktuellen numlock Status.

NUMLOCK ON

Schaltet den numlock ein.

NUMLOCK OFF

Schaltet den numlock aus.

Anforderungen

LasalOS: ab LasalOS 5.50 ist erforderlich.

1.4.3.35 PACKAGE

Mit diesem CLI-Befehl können zusätzliche Funktionen, wie z.B. ein Dateiserver, nachträglich installiert werden.

Für eine Installation benötigen Sie ein SIGMATEK-Installationspaket (ZIP-Archiv) der gewünschten Funktion. Den Inhalt dieses Archivs entpacken Sie in das Verzeichnis C:\lslsys\packages. Anschließend können Sie das SIGMATEK-Installationspaket mit dem Befehl „PACKAGE INSTALL <Name des Pakets>“ installieren.

Im SIGMATEK-Installationspaket finden Sie eine Datei liesmich.txt. Diese beschreibt den Installationsvorgang des jeweiligen Pakets.



Syntax

PACKAGE [Option] „Kommando“ „Paketname“

Optionen:

- I - Verzögerte Installation des Pakets (nach dem nächsten Reboot)
- d Zeigt Debug-Ausgaben am CLI an

Commands:

- LIST Zeigt eine Liste der installierten Pakete an
- INSTALL Installiert das angegebene Paket
- UNINSTALL Entfernt das angegebene Paket

Beispiele

PACKAGE LIST

Zeigt die Liste aller installierten Pakete an

PACKAGE INSTALL samba

Installiert das Paket Samba

PACKAGE -l INSTALL samba

Installiert das Paket Samba nach dem nächsten Reboot

Anforderungen

Lasal OS Salamander ab Version 09.03.030.

1.4.3.36 PING

Sendet ein Ping-Signal an einen Remote-Host und überprüft die Antwort. 4 Pakete mit je 32 Byte Daten werden übertragen.

Syntax

PING ip address

Die IP-Adresse des Remote-Host (muss in gepunkteter Dezimalschreibweise angegeben werden).

Ein Ping kann benutzt werden um festzustellen, ob eine Maschine über das Netzwerk erreicht werden kann.

Beispiel

PING 192.168.44.106

1.4.3.37 PNPBIOS

Zeigt Informationen über die PnP- und PCI-BIOS.

Syntax

PNPBIOS

1.4.3.38 QUCMD

Legt einen Befehlsstring in eine Warteschlange, der nach der Ausführung der autoexec.lsl verarbeitet wird. Dieser Befehl hat keine Wirkung, wenn er außerhalb der autoexec.lsl aufgerufen wird.

Syntax

QUCMD Befehlsstring

1.4.3.39 REBOOT

Setzt und startet die Maschine neu.

Syntax

REBOOT

1.4.3.40 REXX

Führt ein REXX-Programm aus.

Syntax

REXX [program-name]

Programmname ist der Name des Rexx-Programms, das ausgeführt werden soll. Wenn kein Programm angegeben ist, wartet LasalOS-Rexx auf den Eingabe von Rexx-Befehlen und führt die Befehle aus, wenn das Ende-der-Datei-Zeichen (Strg-Z) eingegeben wird.

Bemerkungen

Siehe Dokumentation [LasalOS Rexx](#) für eine Beschreibung der REXX.

1.4.3.41 ROUTE

Zeigt und manipuliert die Routing-Tabelle.

Syntax

PRINT

Zeigt die Einträge in der Routing-Tabelle.

ADD

Fügt einen Eintrag in der Routing-Tabelle ein.

DEL

Löscht einen Eintrag aus der Routing-Tabelle.

Die Routing-Tabelle besteht aus 5 Spalten. Die erste Spalte enthält die Zieladresse des Netzwerks, die zweite enthält die dazugehörige Netzwerkmaske. Der dritte Eintrag ist die Gateway-Adresse; das ist die Ziel-Adresse dieses Pfads. Die vierte ist die Netzwerk-Schnittstelle (im IP-Adresse Format) und die letzte ist Metrik.

Um einen Eintrag in der Routing-Tabelle einzufügen, müssen diese 5 Einträge gesetzt werden. Um einen Eintrag zu löschen muss nur die Zieladresse des Netzwerks und die Netzmaske angegeben werden.

1.4.3.42 RTTASKS

Zeigt alle Real-Time Tasks der Applikation.

Syntax

RTTASKS

1.4.3.43 SEMAS

Zeigt eine Liste der bestehenden Flags.

Syntax

SEMAS

SEMAS wird die Flag-Namen, Typen, Werte zeigen sowie eine Liste aller Tasks, die auf den jeweiligen Flag warten. Für Ressource- und Mutex-Flags, wird der Name der aktiven Task (falls vorhanden) ebenfalls zurückgegeben.

1.4.3.44 SET CYCLIC

Zeigt oder setzt das Zeitfenster für einen Task niedriger Priorität.



Sie sollten die Cyclic-Task-Delay-Zeit nur andern, wenn Sie wirklich wissen, was Sie tun!

Syntax

SET CYCLIC

Zeigt das zyklische Task Zeitfenster.

SET CYCLIC DELAY value

Stellt das zyklische Task Zeitfenster ein (Wert: Zeitfenster des zyklischen Tasks in ms).

1.4.3.45 SET DBGHEAP

Setzen/löschen der Debug Heap-Einstellungen. Diese Einstellungen werden erst nach einem Reset-Run wirksam.

Diese Einstellungen dienen nur zur Fehlersuche. Sie können helfen, Speicherfehler zu finden wie

- Realloc oder Free mit einem ungültigen Zeiger oder einem Zeiger auf einen Speicherblock, der bereits freigegeben wurde.
- Nicht genug Speicher vorhanden (wenn dieser Fehler von der Applikation nicht behandelt wird!).
- Eine Applikation, die nach realloc() den alten Zeiger verwendet anstatt den von realloc() zurückgelieferten.
- Schreiben vor oder nach einem zugewiesenen Speicherblock.
- Schreiben in einen bereits freigegebenen Speicherblock

Da der Application-Heap bei jeder Heap-Operation (Malloc, Realloc, Free) auf Fehler überprüft wird, können Heap-Operationen recht zeitaufwendig sein. Wenn die Applikation viele Heap-Operationen verwendet, kann die Applikation erheblich verlangsamt werden und es kann möglicherweise zu Laufzeitfehlern kommen. Insbesondere bei der Verwendung der Visualisierung kann der Wechsel von Seiten/Bildern langsam sein.

Wird der Heap ausgiebig genutzt, empfiehlt es sich, während der Entwicklung einer Applikation SET DBGHEAP ALL ON zu verwenden. Ist die Entwicklung abgeschlossen und steht fest, dass keine weiteren Speicherfehler vorliegen, sollte die Heap-Prüfung für die Produktion deaktiviert werden.

Syntax

SET DBGHEAP

Anzeige der aktuellen DBGHEAP-Einstellungen.

SET DBGHEAP CHECK_ALWAYS ON|OFF

ON:Der gesamte Applikations-Heap wird bei jedem Heap-Funktionsaufruf auf Integrität geprüft.

OFF - Nur der in ReAlloc() oder Free() angegebene Block wird entsprechend den anderen DBGHEAP Einstellungen geprüft.

SET DBGHEAP ERR_OUTOFMEM ON|OFF

ON:Das OS löst einen AppMem-Fehler aus, wenn Malloc() oder ReAlloc() wegen eines Speicherwegs fehlschlägt.

SET DBGHEAP REALLOC_NEW_PTR ON|OFF

ON:Die OS_SSR_ReAlloc Funktion reserviert einen neuen Block immer an einer anderen Adresse als der des ursprünglichen Blocks.

SET DBGHEAP CHECK_LIMIT_BEGIN ON|OFF

ON:Der Heap-Manager reserviert einen Block, der größer ist als die geforderte Größe, um den Bereich vor den eigentlichen Daten auf Überschreibungen zu kontrollieren. Eine 4 Byte Pufferzone vor dem Datenbereich wird unter RTOS mit 16#FD und unter Salamander mit 16#FC befüllt. Das OS löst bei der Freigabe dieses Blocks mit Free() oder bei jedem Heap-Funktionsaufruf (siehe CHECK_ALWAYS ON|OFF) einen AppMem-Fehler aus, wenn das Bitmuster in der Pufferzone überschrieben wurde.

SET DBGHEAP CHECK_LIMIT_END ON|OFF

ON:Der Heap-Manager reserviert einen Block, der größer ist als die geforderte Größe, um den Bereich nach den eigentlichen Daten auf Überschreibungen zu kontrollieren. Eine 4 Byte Pufferzone vor dem Datenbereich wird unter RTOS mit 16#FD und unter Salamander mit 16#FC befüllt. Das OS löst bei der Freigabe dieses Blocks mit Free() oder bei jedem Heap-Funktionsaufruf (siehe CHECK_ALWAYS ON|OFF) einen AppMem-Fehler aus, wenn das Bitmuster in der Pufferzone überschrieben wurde.

SET DBGHEAP FILL_FREE_BLOCK ON|OFF

ON:Der Heap-Manager füllt freigegebene Blöcke mit 0xDD. Damit kann die Applikation prüfen, ob weiterhin in den freigegebenen Speicher geschrieben wird.

SET DBGHEAP FILL_NEW_BLOCK ON|OFF

ON:Neue Blöcke werden beim Zuordnen mit 0xCD gefüllt.

SET DBGHEAP DBGHEAP STORE_IP ON|OFF

ON:Der Heap-Manager reserviert einen Block, der größer ist als die geforderte Größe, um in dem zusätzlichen Bereich die Instruction Pointer (IP) der Funktionen zu speichern, die die Heap-Funktion aufgerufen haben. Im Falle eines AppMem-Fehlers werden die IP des fehlerhaften Blocks in das Systemprotokoll geschrieben.

SET DBGHEAP CHECK_FREE_PTR ON|OFF

ON:Das OS löst einen AppMem-Fehler aus, wenn bei Free() ein Speicherblock angegeben wird, der bereits freigegeben wurde.

SET DBGHEAP ALL ON|OFF

ON:Aktiviert alle der oben genannten DBGHEAP-Einstellungen.

OFF:Deaktiviert alle der oben genannten DBGHEAP-Einstellungen.

SET DBGHEAP TRIGGERADDR <addr>

Ist <addr> ungleich Null, schreiben die Funktionen MALLOC und REALLOC den Call-Stack in die Event-Log Datei. Dies ist beispielsweise dann nützlich, wenn die Adresse eines beschädigten Speicherblocks bekannt ist und die Funktion ermittelt werden soll, die ihn reserviert hat. Der Code kann dann untersucht werden um festzustellen, was mit dem Speicherblock passiert.

Anforderungen

Die Debug-Heap Funktionen sind ab LasalOS 5.52 verfügbar.

Die Einstellung TRIGGERADDR ist ab LasalOS 1.1.51 vorhanden.

1.4.3.46 SET DEBUG

Weist LasalOS an, welcher Debugger verwendet wird.

Syntax

SET DEBUG LASAL

Lasal Klasse-Debugger zum Debuggen von Applikationen (Standard).

SET DEBUG MSDEV|RTD32

Microsoft VC oder Borland Debugger wird verwendet.

1.4.3.47 SET ERRORLOG

Zeigt bzw. aktiviert/deaktiviert die Kernel-Fehlerprotokollierung.

 Ab LasalOS 5.42 ist dieser Befehl veraltet. Verwenden Sie stattdessen [SET EventLog](#).

Syntax

SET ERRORLOG

Zeigt die Einstellungen der Kernel-Fehlerprotokollierung.

SET ERRORLOG ON|OFF

ON aktiviert, OFF deaktiviert die Kernel-Fehlerprotokollierung.

Wird Kernel Fehlerprotokollierung aktiviert, verwenden Sie "[GET ErrorLog](#) ..." um die Log-Datei zu kontrollieren.

1.4.3.48 SET EVENTLOG

Zeigt oder setzt die Parameter zur Protokollierung der Nachrichten des Betriebssystems.

Syntax

SET EVENTLOG

Zeigt die Einstellungen der Protokollierung der Nachrichten des Betriebssystems.

SET EVENTLOG ON [file-quota]

Aktiviert die Protokollierung der Nachrichten des Betriebssystems in einem Nachrichtenpuffer. Der Parameter file-quota kann verwendet werden, zur Aktivierung oder Deaktivierung des Schreibens des Betriebssystem-Nachrichtenpuffers auf eine Diskette-Datei und um die maximale Größe dieser Datei festzulegen. Das file-quota kann auf einen der folgenden Werte gestellt werden, der Standardwert ist -1:

Wert	Bedeutung
0	Deaktiviert das Schreiben des Betriebssystem-Nachrichtenpuffers in einer Log-Datei.
-1	Ermöglicht das Schreiben des Betriebssystem-Nachrichtenpuffers in einer Log-Datei, ein System-Standardwert wird für die maximale Größe der Log-Datei verwendet.
>0	Ermöglicht das Schreiben des Betriebssystem-Nachrichtenpuffers in einer Log-Datei, ein System-Standardwert wird für die maximale Größe der Log-Datei verwendet.

Der Name der Betriebssystem-Nachrichtendatei ist EVENT00.LOG. Der Standardpfad für diese Datei ist C:\SYSMSG. Dieser Pfad kann überschrieben werden, indem Sie die Umgebungsvariable SYSMSGPATH auf einen anderen Wert setzen. Um die Umgebungsvariable zu setzen, verwenden Sie den CLI Befehl [SETENV](#).

Sehen Sie das [LasalOS Benutzerhandbuch](#) für eine Beschreibung von Berücksichtigungen zum Schreiben von Betriebssystem-Nachrichten an eine Nachrichtendatei (Kapitel sysmsg Facility).

SET EVENTLOG OFF

Deaktiviert die Protokollierung der Nachrichten des Betriebssystems an einen Nachrichtenpuffer.

Bemerkungen

Die Standardeinstellung ist SET EVENTLOG OFF.

Anforderungen

LasalOS: LasalOS ab V 5.42 wird benötigt.

1.4.3.49 SET SYSTRACE

Aktiviert bzw. deaktiviert die SPS-Trace Einrichtung.

Syntax

SET SYSTRACE [start-condition stop-condition type-mask]

start-condition und stop-condition sind Bitmasken, wobei ein Bit ein bestimmtes Ereignis angibt, das die Start- oder Stopp-Bedingung für den Trace auslösen soll. Eine bitweise ODER-Kombination der folgenden Werte ist möglich (das Präfix 0x bedeutet Hexadezimal-Format):

Wert	Bedeutung
0x00000001	Run
0x00000002	Reset
0x00000004	Fehler (z. B. Access Exception)
0x00000008	CDias Watchdog
0x00000010	Runtime Error
0x00000020	Tastatur- oder Maus-Event
0x00000040	Reserviert für den LSE-Kernel
0x00000080	Reserviert für den LSE-Kernel
0x00000100	Reserviert für Applikations-Events
0x00000200	Reserviert für Applikations-Events
0x1 <n> 000000	Event nach n Sekunden
0x40000000	Sofort
0x80000000	Trace-Puffer voll

type-mask ist eine Bit-Maske, bei der ein Bit die Art des Trace-Eintrags angibt, der in den Trace-Puffer geschrieben werden soll. Eine bitweise ODER-Kombination der folgenden Werte ist möglich (das Präfix 0x bedeutet Hexadezimal-Format):

Wert	Bedeutung
0x0001	Task-Schalter
0x0002	Änderung der Applikationszustand (Run, Reset, Fehler)
0x0004	Dias events

0x0008	Aufruf von zyklischen Applikationsmethoden (rtwor, cywork, Background)
0x0010	Reserviert für Interrupt-Events
0x0020	Reserviert für CAN-Events
0x0040	Tastatur- oder Maus-Event
0x0080	LSE-Kernel Event Typen 0
0x0100	LSE-Kernel Event Typ 1
0x0200	User Event Typen 0
0x0400	User Event Typen 0

Wenn die Start- und Stopp-Bedingungen erfüllt sind, wird der Trace-Puffer in die Datei C:\SYMSMSG\TRACE00.CSV geschrieben. Diese Datei kann mit der Lasal2 PLC-Trace-View angezeigt werden. Nachdem der Trace-Puffer in eine Datei geschrieben wurde, wird die Startbedingung zurückgesetzt und muss wieder aktiviert werden, wenn ein weiterer Trace aktiviert werden soll.

Die Parameter können in Dezimal- oder im Hexadezimal Format eingetragen werden. Das Präfix 0x spezifiziert Hexadezimal-Format.

Die Größe des Trace-Puffers kann mit dem Befehl SET TRACEBUFSIZE eingestellt werden.

Bitte beachten Sie, dass ein laufender Trace das Laufzeitverhalten beeinflussen kann. Seien Sie daher vorsichtig, wenn Sie den Trace in einer Produktionsumgebung aktivieren.

Beispiele

```
SET SYSTRACE 1 0x10 0xFFFF
```

Der Trace beginnt bei RUN und endet, wenn ein Laufzeitfehler auftritt. Alle Ereignisse werden in den Trace-Puffer geschrieben.

```
SET SYSTRACE 0x40000000 0x80000000 0x0001
```

Der Trace startet sofort und endet, wenn der Puffer voll ist. Nur Ereignisse vom Typ Task-Switch werden in den Trace-Puffer geschrieben.

Anforderungen

LasalOS: ab 01.01.086 wird benötigt.

1.4.3.50 SET TRACEBUFSIZE

Setzt die Größe des Puffers in Byte, der für die SPS-Trace Einrichtung verwendet wird.

Syntax

```
SET TRACEBUFSIZE size
```

Betriebssystem Salamander: TRACEBUFSIZE kann nur in der AUTOEXEC.LSL eingestellt werden. Die Größe muss zwischen 128 kB und 3 MB eingestellt werden.

1.4.3.51 SET USB_CACHE_FLUSHING

Aktiviert oder deaktiviert das explizite Flushing der Schreibcaches auf angeschlossenen USB-Sticks über zusätzliche USB-Transaktionen oder zeigt den momentanen Betriebszustand dieses Mechanismus an.

USB-Sticks besitzen einen internen, flüchtigen Speicher, in dem per USB empfangene Daten abgelegt werden, bevor sie von der Firmware des USB-Sticks in den nicht-flüchtigen Flash-Speicher geschrieben werden. Wird der USB-Stick vom USB und damit von seiner Spannungsversorgung getrennt, bevor der Inhalt dieses Caches vollständig ins Flash geschrieben worden ist, tritt ein Datenverlust auf. Unter LasalOS soll es jedoch einem Nutzer möglich sein, einen USB-Stick abzustecken ohne vorher – wie z.B. unter diversen Desktop-Betriebssystemen – das OS über das bevorstehende Abstecken zu informieren bzw. den Stick auszuwerfen. Daher wird von LasalOS nach bestimmten Dateioperationen standardmäßig ein Transfer der im Schreibcache eines USB-Sticks vorhandenen Daten ins Flash angestoßen, indem ein bestimmtes Kommando an die Firmware des Sticks geschickt wird. Die Verwendung dieses Kommandos führt in der Regel zu keinen bzw. unerheblichen Einbußen in den erzielten Datenraten bei Schreibzugriffen auf einen USB-Stick. Für den Fall, dass zukünftig bei einem bestimmten USB-Stick-Modell zu niedrige Datenraten gemessen werden, kann der beschriebene Mechanismus mit dem Befehl SET USB_CACHE_FLUSHING OFF deaktiviert werden. Aufgrund des dadurch entstehenden, je nach USB-Stick-Modell unterschiedlich großen Risikos eines Datenverlustes sollte dies jedoch nur vorübergehend bzw. zu Testzwecken geschehen.

Die Standardeinstellung ist SET USB_CACHE_FLUSHING ON.



Syntax

```
SET USB_CACHE_FLUSHING [ON|1|OFF|0|HELP]
```

```
SET USB_CACHE_FLUSHING
```

Gibt den aktuellen Betriebszustand des Cache-Flushing-Mechanismus aus.

```
SET USB_CACHE_FLUSHING ON oder SET USB_CACHE_FLUSHING 1
```

Aktiviert explizites Flushing der Schreibcaches des angeschlossenen USB-Sticks.

```
SET USB_CACHE_FLUSHING OFF oder SET USB_CACHE_FLUSHING 0
```

Deaktiviert explizites Flushing der Schreibcaches des angeschlossenen USB-Sticks.

```
SET USB_CACHE_FLUSHING HELP
```

Gibt Informationen zu Anwendung und Wirkung dieses Befehls aus.

Anforderungen und Verfügbarkeit

Der Befehl SET USB_CACHE_FLUSHING ist ab LasalOS 01.03.070 auf den Plattformen C-IPC, ETV und ETV-EDGE verfügbar.

1.4.3.52 SETENV

Setzt oder löscht den Wert einer Umgebungsvariable.



Verwenden Sie den Befehl [SHOWENV](#), um den Wert einer Umgebungsvariable anzuzeigen.

Syntax

```
SETENV variable-name
```

Dieser Befehl löscht die Umgebungsvariable.

```
SETENV variable-name wert
```

Dieser Befehl setzt den Wert einer Umgebungsvariable.

Anforderungen

LasalOS: LasalOS ab V 5.42 wird benötigt.

1.4.3.53 SETLEDSTATUS

Setzen eines LED-Status.

Syntax

```
SETLEDSTATUS led status
LED - RUN, STATUS, ERROR
status - OFF, ON, FLASH
```

Anforderungen

LasalOS ab Salamander 09.01.140

1.4.3.54 SHOWENV

Zeigt den Wert einer Umgebungsvariable.



Verwenden Sie den Befehl [SETENV](#), um den Wert einer Umgebungsvariable einzustellen oder zu löschen.

SyntaxS

```
SHOWENV variable-name
```

Anforderungen

LasalOS: ab 5.42 wird benötigt.

1.4.3.55 SRAMLOAD

Setzt die SRAM aus einer Datei wieder zurück, die zuvor mit SRAMSAVE erstellt wurde.

Syntax

```
SRAMLOAD filename
```

Anforderungen

LasalOS: ab 1.1.8 wird benötigt.

1.4.3.56 SRAMSAVE

Speichert den Inhalt des SRAM in eine Datei.

Syntax

SRAMSAVE Dateiname

Anforderungen

LasalOS: ab 1.1.8 wird benötigt.

1.4.3.57 STATUS

Zeigt die LASAL OS Statusinformationen.

Syntax

STATUS

1.4.3.58 TASKS1/TASKS

Zeigt Informationen über die aktuell laufenden Prozesse.

Kurzform: TASKS

Syntax

TASKS1

1.4.3.59 TASKS2

Zeigt Informationen über die aktuell laufenden Prozesse und ihre Positionen, in denen sie warten.

Syntax

TASKS2

1.4.3.60 UNMOUNT

Einen USB-Stick ordnungsgemäß entfernen.

Syntax

```
UNMOUNT <device letter>
```

Beispiel

```
UNMOUNT E
```

Unmounts the Device E:\

1.4.3.61 VER

Zeigt die LASAL OS-Version mit dem Freigabedatum, die aktuelle Auflösung und Kontaktinfos. Zusätzlich kann dieser Befehl dazu verwendet werden, um die Version einer DLM anzuzeigen. Hierzu muss lediglich ein optionaler Übergabeparameter `ddlmname.dlm` übergeben werden.

Syntax

```
VER  
VER -D zlib.dlm
```

1.4.3.62 vpnd daemon

Der VPN-Client-Dienst kann durch den (Remote-)CLI-Befehl manuell gestartet oder gestoppt werden. Der optionale Parameter `<configuration>` ist der Name der VPN-Konfigurationsdatei ohne die Endung `.conf`. Die VPN-Konfigurationsdateien müssen sich im Verzeichnis `C:\lslsys\vpn` befinden. Der Name einer VPN-Konfigurationsdatei muss die Endung `.conf` aufweisen.

Der Befehl ist nur auf Edge2-CPU's mit Salamander-OS ab Version 09.03.100 verfügbar.



Syntax

```
vpnd daemon start [<configuration>] | stop [<configuration>] | restart [<configuration>] | reload [<configuration>]
```

Beispiele

```
vpndemon start
```

Startet einen VPN daemon für jede VPN-Konfigurationsdatei im Verzeichnis C:\lslsys\vpn.

```
vpndemon start <configuration>
```

VPN daemon mit der VPN-Konfigurationsdatei <configuration>.conf starten. Die VPN-Konfigurationsdatei muss sich im Verzeichnis C:\lslsys\vpn befinden.

```
vpndemon stop
```

Alle laufenden VPN daemons stoppen.

```
vpndemon stop <configuration>
```

Sucht einen laufenden VPN daemon, der mit der angegebenen VPN-Konfigurationsdatei gestartet wurde, und stoppt ihn. Die VPN-Konfigurationsdatei muss sich im Verzeichnis C:\lslsys\vpn befinden.

```
vpndemon restart
```

Stoppt alle laufenden VPN daemons und startet einen VPN daemon für jede VPN-Konfigurationsdatei im Verzeichnis C:\lslsys\vpn.

```
vpndemon restart <configuration>
```

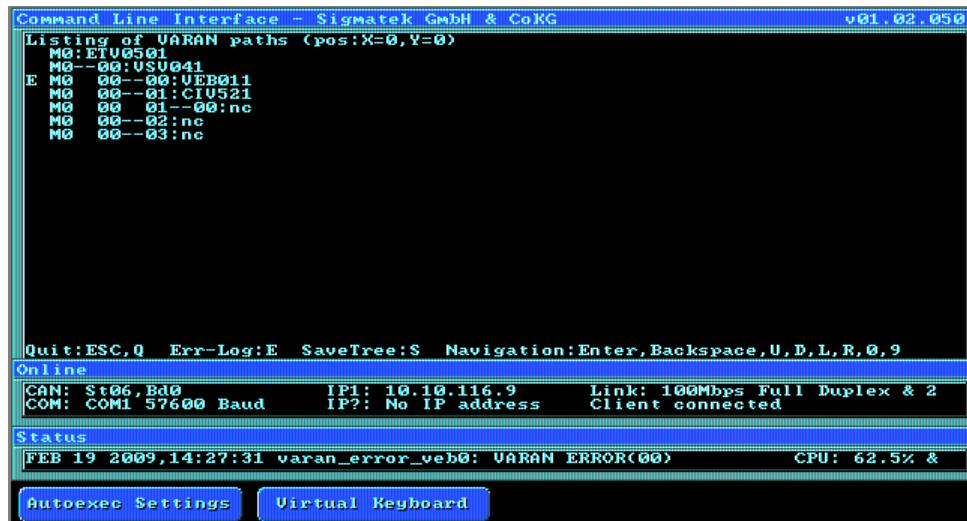
Sucht einen laufenden VPN daemon, der mit der angegebenen VPN-Konfigurationsdatei gestartet wurde, und stoppt und startet ihn erneut. Die VPN-Konfigurationsdatei muss sich im Verzeichnis C:\lslsys\vpn befinden.

```
vpndemon reload
```

Führt für alle laufenden VPN daemons einen Restart aus. Wenn ein VPN daemon nicht unter dem Benutzer root läuft, muss eine passende VPN-Konfigurationsdatei im Verzeichnis C:\lslsys\vpn vorhanden sein.

1.4.3.63 VT

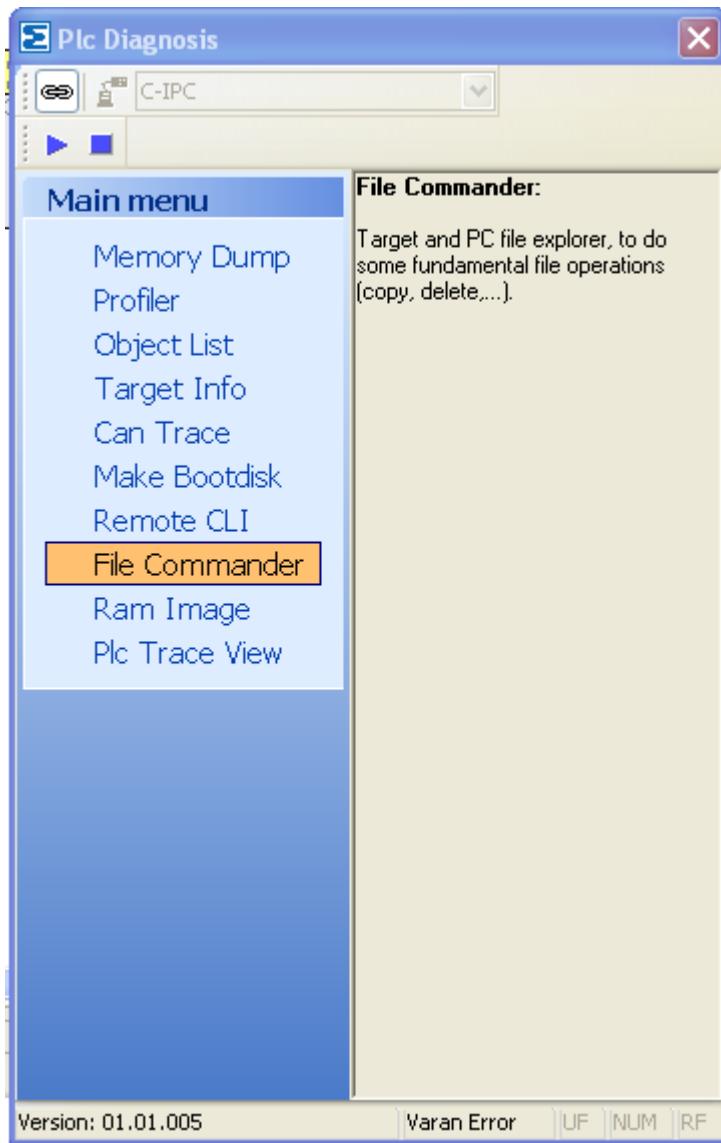
Durch die Anzeige des Fehlerbaums mit VT wird der folgende Bildschirm wird angezeigt. Der Fehler wird mit E identifiziert. Hier kann die SaveTree:S Taste benutzt werden, um den gesamten Baum im Verzeichnis C:\SYSMSG\ in der ErrorTree.txt Datei zu speichern.



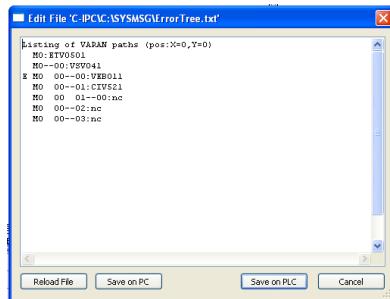
Liste der VARAN-Pfade (pos: X = 0, Y = 0)

M0:ETV0501
M0--00:VSV041
E M0 00--00:VEB011
M0 00--01:CIV0521
M0 00 01--00:nc
M0 00--02:nc
M0 00--03:nc

Die Datei kann mit einer Online-Verbindung gelesen werden.



C:\SYSMSG\ErrorTree.txt



1.4.4 Projektbefehle

1.4.4.1 SET FPUROUNDCONTROL

Ändert den Rundenmodus der FPU

Nur für Benutzer, die über entsprechende FPU-Kenntnisse verfügen!



Syntax

set fproundcontrol x

Wert zwischen 0 und 3

0 Runde auf die nächste

1 Abrunden

2 Aufrunden

3 Kürzen

Anforderungen

LasalOS: OS Version >01.02.051

1.4.4.2 LSLLOAD/LO

Lädt ein gespeichertes Projekt von der Festplatte.

Kurzform: LO

Syntax

```
LSLLOAD [drive:][path]
```

Zur einfachen Verwendung ist der Standard-Übertragungspfad c:\LSLWORK\PROJECT.IDX.



1.4.4.3 LSLSAVE/SA

Speichert das aktuelle Projekt auf der Festplatte.

Zur einfachen Verwendung ist der Standard-Pfad zum Speichern c:\LSLWORK\PROJECT.IDX. Wurde ein Name angegeben, dann wird die Index-Datei zur Verwendung mit LSLLOAD ohne einen bestimmten Pfad als PROJECT.IDX dupliziert.

Wird die Option NORESET angegeben und die Applikation läuft, wird das Projekt gespeichert, ohne die Applikation zu stoppen bzw. einen Reset auszulösen. Während des Speicherns können Latenzen auftreten. Diese Option kann in Verbindung mit LasalClass2/PLC Diagnosis (Remote CLI) erst mit den Versionen 2.2.157 (LC2), bzw. 1.1.041 (PLCDiag) benutzt werden. Versionen davor lösen automatisch einen Reset aus.

Bei einem negativen Rückgabewert finden sie weitere Informationen in der Liste am Ende des Dokuments.

Kurzform: SA

Syntax

```
LSLSAVE [drive:][path][NORESET]
```

1.4.4.4 RUN

Versucht die LASAL Applikation durchzuführen.

Wird RUN innerhalb der autoexec.lsl ausgeführt und Online-Kommunikation ist vorhanden, bleibt die Applikation im Reset-Modus.

Wird RUN innerhalb der autoexec.lsl ausgeführt und das Argument UC zugefügt, wird eine Online-Kommunikation ignoriert und die Applikation wird gestartet.

Beim Starten der Anwendung aus der Command Line Interface, haben RUN und RUN UC das gleiche Verhalten.

Syntax

RUN
RUN UC

1.4.4.5 RESET

Setzt das LASAL OS in den Reset-Modus.

Syntax

RESET

1.4.4.6 PROJECT

Zeigt die Projektinformationen.

Syntax

PROJECT command
PROJECT MODULES

Zeigt die Projektmodul-Liste.

PROJECT LINK

Verknüpft das aktuelle Projekt.

PROJECT ERRORS

Zeigt Projekt Link-Fehler.

PROJECT INFO

Zeigt Informationen zum Projekt.

PROJECT SYMBOL symbol

Zeigt Projekt Symbolinformationen.

PROJECT SAVE [directory]

Speichert das Projekt im angegebenen Verzeichnis auf der Festplatte.
(Standard Verzeichnis C:\LSLWORK)

PROJECT LOAD [directory]

Lädt das Projekt vom angegebenen Verzeichnis auf der Festplatte.
(Standard Verzeichnis C:\LSLWORK).

PROJECT CLEAR

Löscht das aktuelle Projekt aus dem Speicher.

1.4.5 Konfigurationsbefehle

1.4.5.1 ADDBR

Dieser Befehl erstellt ein Bridge Netzwerk Interface. Er steht nur beim Betriebssystem Salamander ab der Version 09.03.177 zur Verfügung.

Mit einer Bridge können mehrere physikalische Netzwerke an mehreren IP-Schnittstellen zu einem gemeinsamen Netzwerk zusammengefasst werden, das dann an einem logischen Bridge Netzwerk Interface angeschlossen ist.

Die zusammengefassten Netze haben dabei die gleiche Subnetzmaske. Die Steuerung, auf der die Bridge konfiguriert wurde, arbeitet dann ähnlich wie ein Switch.

Die Bridge stellt sich als IP50 dar. Man kann ihr eine IP-Adresse zuweisen, damit das Gerät, auf dem sich die Bridge befindet, ansprechbar ist. Die Interfaces, die in die Bridge gegeben werden, haben dann keine IP-Adresse mehr.

Syntax

```
ADDBR 50 <Liste der Interfaces die hinzugefügt werden sollen> [stp]
```

Wenn stp angegeben ist, wird das Spanning Tree Protokoll aktiviert (Standard ist nicht aktiv).

Mit dem Spanning Tree Protokoll werden unerwünscht kreisende Pakete verhindert.

Da das Spanning Tree Protokoll eine erhöhte CPU-Last bewirkt, sollte darauf geachtet werden, dass die Netzwerktopologie so gestaltet wird, dass keine Schleifen möglich sind und auf das Spanning Tree Protokoll verzichtet werden kann.

Beispiel

Zusammenfassung eines WLAN- mit einem Kabelnetz.

```
ADDBR 50 1 10 11
```

fügt IP1, IP10 und IP11 zur Bridge IP50 hinzu

Zusammenfassung zweier Ethernet-Schnittstellen

```
ADDBR 50 1 2
```

fügt IP1 und IP2 zur Bridge IP50 hinzu

Der Bridge kann man dann auch noch eine IP-Adresse geben

```
SET IP 50 HOSTADDR <ipaddr>
```

```
SET IP 50 SUBNET <subnet>
```

1.4.5.2 CALIB

Startet ein Programm zur Kalibrierung des Touch.

Syntax

CALIB

Für OS-Versionen \geq 1.1.214:

CALIB [nx ny]

nx - die Zahl der Kalibrierungspunkte in der x-Richtung

ny - Die Zahl der Kalibrierungspunkte in der y-Richtung

Folgen Sie den Anweisungen und die Kalibrierungsdaten werden im Hauptverzeichnis in der TOUCHCFG.DAT Datei gespeichert.

Hinweise

Im Hinblick auf die Verwendung der virtuellen Touch-Tastatur (z.B. in einer C-IPC CLI), muss eine gültige Touch-Schnittstelle konfiguriert werden, sowie die *.SML-Dateien im Verzeichnis C:\lssys platziert werden. Diese Dateien sind auf der Master-CF-Karte zu finden.

1.4.5.3 Calib check

Dieser Befehl ist nur mit dem Salamander-Betriebssystem verfügbar. Er muss in der Datei autoexec.lsl eingetragen werden.

Beim Hochlauf der Steuerung prüft dieser Befehl, ob der Touch-Screen bereits kalibriert wurde. Wenn die Datei C:\touchdat.cfg nicht vorhanden ist, dann wird die Kalibrierung des Touch-Screen gestartet. Siehe Befehl [CALIB](#).

1.4.5.4 CHANGECERT

Syntax

CHANGECERT option <filename>

Option	Beschreibung
-s/-svr	Ändern des Server-Zertifikats und der Schlüsseldatei
-c/-cli	Ändern des Client-Zertifikats

Beispiele

```
CHANGECERT -svr c:/path/to/cert c:/path/to/key
```

Für Server.

```
CHANGECERT -cli c:/path/to/cert
```

Für Clients.

1.4.5.5 DIAS

Zeigt die verfügbaren DIAS-Stationen.

Syntax

```
DIAS [nr]
```

nr - Die Zahl der DIAS-Stationen

Außerdem ist ein DIAS-MASTER vorhanden, werden auch diese Funktionen zur Verfügung gestellt:

```
DIAS UPDATE filename sector
```

Aktualisiert das DIAS-Master Programm. Vor der Aktualisierung wird eine Backup-BIN Datei mit dem Inhalt des Flash im \LSLSYS\Systemverzeichnis mit einem eindeutigen Dateinamen erstellt.

filename - Die zu aktualisierende HEX (MCS) oder BIN-Datei

sector- Die Nummer des zu aktualisierenden Flash-Bereichs (0 oder 1)

```
DIAS BACKUP filename sector
```

Speichert den aktuellen Inhalt eines Flash-Sektors in einer BIN-Datei, die zum Wiederherstellen von älteren Versionen verwendet werden kann.

filename - Der Name der Backup-BIN Datei

sector - Die Nummer der Flash-Sektor-Backup (0 oder 1)

```
DIAS VERIFY filename Sektor
```

Vergleicht einer bestimmten Datei und den Inhalt eines Flash-Sektors.

filename - Die zu vergleichende HEX (MCS) oder BIN-Datei

sector - Die Anzahl der zu vergleichenden Flash-Bereiche (0 oder 1)

```
DIAS INFO
```

Zeigt den Dateinamen der aktuellen Master-Programme in den Flash-Sektoren.

1.4.5.6 EURO

Setzt der Anzeige des Datums bzw. der Uhrzeit auf des europäischen Format.

Syntax

EURO

Um wieder auf das US-Format zu wechseln, verwenden Sie der [US](#)-Befehl.

1.4.5.7 FIREWALL

Dieser Befehl dient zum Sperren und Freigeben von Ports.



Er ersetzt auf keinen Fall eine professionelle vorgeschaltete Firewall. Wir raten außerdem ausdrücklich davon ab, eine Steuerung ohne zwischengeschaltete professionelle Firewall mit dem Internet zu verbinden.

Syntax

`FIREWALL <command> <option> <option>`

Kommandos

LIST	Listet die konfigurierten Regeln im iptables-Format auf
LISTLONG	Listet die konfigurierten Regeln im Detail und Plaintext-Format auf
BLOCKALL	Alles ist blockiert, außer der Loopback-Vorrichtung
SIGSTD	Alles ist blockiert, außer der Loopback-Vorrichtung und den SIGMATEK standard ports Offenen Input-Ports (TCP)
1954	LASAL Kommunikation
1955	Comlink
1956	Comlink refresh list
2054	LASAL Kommunikation (SSL)
2055	Comlink (SSL)
2056	Comlink refresh list (SSL)
2402	Dataservice loader
9980	Data service

	Offenen Input-Ports (UDP)
	1954 LASAL IP scanner
	Andere Protokolle
	ICMP (PING)
REMOVEALL	Entfernt alle konfigurierten Regeln
REMOVEBYNUMBER	Entfernt eine Regel nach Kette und Nummer
SAVE	Speichert die aktuelle Konfiguration
LOAD	Lädt eine gespeicherte Konfiguration
STARTACCEPT	Wenn Sie die Konfiguration der Ports selbst vornehmen möchten, MUSS dieser Befehl nach dem Befehl BLOCKALL EINMAL aufgerufen werden. Verwenden Sie diesen Befehl nicht, wenn Sie zusätzliche Regeln zur SIGMATEK-Standardkonfiguration konfigurieren möchten.
ACCEPTPORT	Öffnet einen Port für eine oder alle Ethernet-Schnittstellen
ACCEPTADDRESS	Akzeptiert alles von einer definierten IP-Adresse an für eine oder alle Ethernet-Schnittstellen
NO COMMAND	Verwendet die Syntax von IPTABLES

Beispiele

- Die Ports müssen in der Datei autoexec.lsl konfiguriert werden.
- Die Ports sollten konfiguriert werden, bevor die Ethernet-Schnittstellen aktiviert wird (SET IP ...)
- Die Ports können nicht mit dem "ServiceProvider" konfiguriert werden
- !!! Eine Regel nicht durch die Definition einer Gegenregel ineffizient machen, sondern diese Regel löschen!!!

Beispiel 1 (SIGMATEK Standard)

```
FIREWALL REMOVEALL
FIREWALL SIGSTD
```

Beispiel 2 (Erlaubt es einem VNC-Client, sich über alle Schnittstellen zu verbinden – zusätzlich zu SIGSTD)

```
FIREWALL REMOVEALL
```

```
FIREWALL SIGSTD  
FIREWALL ACCEPTPORT 5900
```

Beispiel 3 (Ermöglicht es einem VNC-Client, sich über die IP 1-Schnittstelle zu verbinden – zusätzlich zu SIGSTD)

```
FIREWALL REMOVEALL  
FIREWALL SIGSTD  
FIREWALL ACCEPTPORT 5900 1
```

Beispiel 4 (Alles von der IP-Adresse xx über IP 2 zulassen – zusätzlich zu SIGSTD)

```
FIREWALL REMOVEALL  
FIREWALL SIGSTD  
FIREWALL ACCEPTADDRESS 192.168.0.1 2
```

Beispiel 5 (Selbstkonfiguration ohne Sigmatek-Standard. Nur Lasal und ein vnc-Client dürfen sich verbinden)

```
FIREWALL REMOVEALL  
FIREWALL BLOCKALL  
FIREWALL STARTACCEPT  
FIREWALL ACCEPTPORT 1954  
FIREWALL ACCEPTPORT 5900
```

Beispiel 6 (Entfernen einer EINGABE-Regel)

Entfernt die folgende Regel (LISTLONG)

```
12 0 0 ACCEPT tcp -- eth0 * 0.0.0.0/0 0.0.0.0/0 ctstate NEW tcp dpt:5900  
FIREWALL REMOVEBYNUMBER INPUT 12
```

Beispiel 7 (IPTABLES - allow only 1954 and loopback)

```
FIREWALL "-P INPUT DROP"  
FIREWALL "-A INPUT -i lo -j ACCEPT"  
FIREWALL "-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT"  
FIREWALL "-A INPUT -p tcp -m conntrack --ctstate NEW -m tcp --dport 1954 -j  
ACCEPT"
```

Wenn Sie im Internet nach iptables oder netfilter suchen, finden Sie viele Beispiele.

e.g. <https://netfilter.org/documentation>



1.4.5.8 IP

Zeigt und setzt die aktuellen TCP/IP Einstellungen.

Syntax

IP/IP INFO

Zeigt TCP/IP Informationen; Nummer der Ethernet-Schnittstelle, Ethernet-Chip, Adressen, Link-Status, und bei einem laufenden Server werden die verbundenen Clients angezeigt.

P SHUTDOWN

Beendet den TCP Debug-Server. Stellen Sie sicher, dass alle Clients offline sind. Beachten Sie, dass diese Funktion blockiert, bis der Server komplett abgeschaltet ist. Nach dem Aufruf können keine TCP-Clients verbunden werden, bis ein Neustart erfolgt. Aber die IP-Adresse des Host kann geändert werden, wenn der Server nicht läuft. (Siehe [SET IP](#) ...)

IP RESTART

Startet den TCP Debug-Server nach einer Abschaltung neu.

IP KILLCLIENTS

Killt alle verbundenen Clients.

IP LOOPBACK

Installiert bzw. deinstalliert eine IP Loopback-Schnittstelle.

IP DEBUG

Zeigt die Informationen des TCP-Servers auf dem Bildschirm.

IP LOG

Schreibt die Informationen des TCP-Servers in eine Datei.

IP TCPRETRANS <min> >max>

Zeigt und setzt für die angegebene Ethernet-Schnittstelle die untere und die obere Grenze des TCP Retransmission Timers in Millisekunden. Wenn <min> und/oder <max> Null sind, dann werden die aktuellen Einstellungen angezeigt. Wenn <min> und/oder <max> ungleich Null sind, dann werden die Grenzen auf die angegebenen Werte gesetzt. Die Voreinstellungen sind 20 und 60 000 ms. Dieser Befehl steht nur bei Steuerungen mit dem Betriebssystem RTK zur Verfügung. Dieser Befehl kann während der Laufzeit oder in der Datei Autoexec.lsl ausgeführt werden. Nicht jedoch per Remote CLI.

Mehr als eine Ethernet-Schnittstelle

Alle Befehle können mit den folgenden Zusätzen verwendet werden. Ohne Zusatzangaben, wird immer die erste Ethernet-Schnittstelle verwendet.

IP X (+ Befehl)

X ist die Nummer der Ethernet-Schnittstelle (1 .. erste, 2 .. zweite, ...).

IP ALL (Befehl)

für alle Ethernet-Schnittstellen.

Beispiel

IP SHUTDOWN

Beendet den TCP-Server.

IP RESTART

TCP-Server neu starten.

IP INFO

Zeigt die TCP-Einstellungen (bei mehr Ethernet-Schnittstellen die Daten der ersten).

IP INFO ALL

Zeigt die TCP-Einstellungen aller Ethernet-Schnittstellen.

IP 2 SHUTDOWN

Beendet den TCP Server 2.

IP ALL RESTART

Startet alle TCP-Server neu.

IP 1 TCPRETRANS 0 0

Zeigt für die Ethernet-Schnittstelle 1 die untere und die obere Grenze des TCP Retransmission Timers.

IP TCPRETRANS 0 900

Zeigt für die Ethernet-Schnittstelle 1 die untere Grenze des TCP Retransmission Timers und setzt die obere Grenze auf 900 ms.

1.4.5.9 IPCINFO

Zeigt Informationen über den IPC.

Syntax

IPCINFO

1.4.5.10 SCREENSAVER

Dieser Befehl steht beim Betriebssystem RTK nicht zur Verfügung. Er zeigt oder ändert die Einstellungen des Bildschirmschoners.

Syntax

SCREENSAVER <x>
SCREENSAVER INFO

Optionen

<>>	Zeit von der letzten Berührung des Displays bis zum Aktivieren des Bildschirmschoners in Sekunden. 0 stellt die Voreinstellung von 3600 Sekunden wieder her.
-999	Deaktiviert den Bildschirmschoner. ACHTUNG: Ohne Bildschirmschoner sinkt die Lebensdauer des Displays und es besteht die Gefahr, dass das Display durch längere Zeit dargestellte unbewegte Bilder dauerhaft beschädigt wird (sogenanntes „Einbrennen“).
INFO	Zeigt den eingestellten Wert an.

1.4.5.11 SET

Zeigt und ändert die Betriebssystemoptionen.

Syntax

SET

Zeigt die aktuellen LasalOS-Einstellungen.

1.4.5.12 SET APPCODE

Zeigt oder ändert die Größe des Application Code-Speichers. Der Applikationsdaten-Speicher wird auf den Wert eingestellt: Gesamtspeicher – Application Code-Speicher

RTK	Dieser Befehl kann nicht per Remote-CLI, sondern nur als CLI-Befehl ausgeführt werden (z.B. per USB-Tastatur oder per VNC). Um die Speichergröße zu ändern, kann der Befehl auch in der Datei autoexec.lsl ausgeführt werden.
Salamander	Dieser Befehl kann nicht per Remote-CLI ausgeführt werden. Um die Speichergröße zu ändern, muss der Befehl in der Datei autoexec.lsl ausgeführt werden. Um die Speichergröße anzuzeigen, kann ein CLI-Befehl ausgeführt werden (z.B. per USB-Tastatur oder per VNC). Siehe auch INFO .

Syntax

SET APPCODE size

size – Die Größe der Application Code-Speichers in Kilobyte, minimal 512

Beispiel

SET APPCODE 4096

Diese Befehl setzt den Application Code-Speicher auf 4 MB.

1.4.5.13 SET APPDATA

Zeigt oder ändert die Größe des Applikationsdaten-Speichers. Der Application Code-Speicher wird auf diesen Wert eingestellt: Gesamtspeicher – Applikationsdaten-Speicher.

Dieser Befehl steht nur beim Betriebssystem RTK zur Verfügung. Dieser Befehl kann nicht per Remote-CLI, sondern nur als CLI-Befehl ausgeführt werden (z.B. per USB-Tastatur oder per VNC). Um die Speichergröße zu ändern, kann der Befehl auch in der Datei autoexec.lsl ausgeführt werden.

Syntax

```
SET APPDATA size
```

size Die Größe des Applikationsdaten-Speichers in Kilobyte, minimal 512

Beispiel

```
SET APPDATA 4096
```

Diese Befehl setzt den Application Datenspeicher auf 4 MB.

1.4.5.14 SET APPLSAVE

Konfiguriert den Aufruf von IF_PowerDown (integraler Bestandteil der lse_rtk) bei den folgenden Events.

Das RESET und USV-Event löst den Aufruf IF_PowerDown ohne weitere Beachtung aus.



Syntax

```
SET APPLSAVE DIAS|WATCHDOG|RTRUNTIME 1|0
```

Aktiviert/deaktiviert den Aufruf von IF_PowerDown im Fall eines DIAS-Fehlers, eines Watchdog-Fehlers oder eines Realtime Runtime-Fehlers.

1.4.5.15 SET BOOTTEXT

Ändert den Text für den Applikations-Start.

Wenn SET BOOTTEXT nicht gesetzt ist, wird "Running Loader ..." angezeigt.

Syntax

```
SET BOOTTEXT text
text - Text, der angezeigt werden soll, maximal 47 Zeichen
```

Beispiel

```
SET BOOTTEXT
```

Kein Boottext wird angezeigt

```
SET BOOTTEXT now booting
```

Boottext "now booting" wird angezeigt

1.4.5.16 SET BTRUNTIME

Zeigt oder setzt den Background-Runtime Zähler.

Syntax

```
SET BTRUNTIME
```

Zeigt den Background-Runtime-Zähler.

```
SET BTRUNTIME value
```

Setzt den Background-Runtime-Zähler.

value - die Anzahl der Ticks (1 Tick = 10 ms), Bereich: 0-4294967295, 0 = DISABLED

Ist der Runtime-Zähler aktiviert, dann kontrolliert ein Watchdog die Ausführungszeit des Background Tasks (Background Methode). Wenn die Ausführungszeit des Background Tasks den vom Runtime-Zähler bestimmten Grenzwert überschreitet, stoppt die Anwendung mit dem Status BTRUNTIME.

Standard: 0 = DEAKTIVIERT

1.4.5.17 SET CAN

Zeigt oder ändert die Parameter der CAN-Schnittstelle.

Syntax

```
SET CAN [ifnum [BAUD baudind] [STATION addr]]
```

ifnum - Nummer der CAN-Schnittstelle (1 = CAN1, 2 = CAN2, ...).

baudind - Index zur Baudrate-Tabelle:

- 0 - 615 Kb
- 1 - 500 Kb
- 2 - 250 Kb
- 3 - 125 Kb
- 4 - 100 Kb
- 5 - 50 Kb
- 6 - 20 Kb
- 7 - 1 Mb

addr - die CAN-Stationsnummer (0-31).

SET CAN

Zeigt die Einstellungen für alle CAN-Ports.

SET CAN ifnum BAUD baudind

Setzt die CAN-Baudrate.

SET CAN ifnum STATION addr

Setzt die CAN-Stationsnummer.

SET CAN ifnum ONLCOM mode

Setzt ein Flag, um die Onlinekommunikation via CAN zu sperren oder freizugeben.

mode: 0 - Sperrt die CAN-Onlinekommunikation

1 - CAN-Onlinekommunikation möglich (Defaultwert)

Beispiel

```
SET CAN 1 BAUD 0 STATION 23
```

Dieser Befehl setzt die Baudrate der CAN-Schnittstelle CAN1 auf 615Kb und der Stationsnummer auf 23.

1.4.5.18 SET CLI

Setzt den Command Line Interface (CLI)-Modus.

Syntax

```
SET CLI mode OFF|ON
```

Der Modus wird auf RUN, RESET oder ERROR eingestellt

RUN Öffnet bzw. schließt die CLI beim RUN (Default OFF). Der Wert ON wird nur dann sinnvoll, wenn das laufende Projekt keine Ausgabe auf den Bildschirm zeigt, da die CLI-Ausgabe die Projekt-Ausgabe stören wird.

RESET Öffnet bzw. schließt die CLI beim Reset (Default ON).

ERROR Öffnet bzw. schließt die CLI bei einer EXCEPTION (Default ON).

Beispiel

```
SET CLI RUN ON
SET CLI RESET OFF
SET CLI ERROR ON
```

1.4.5.19 SET CPUBT

Dieser Befehl gibt an, auf welchem CPU-Kern der Background-Task laufen soll.

Syntax

```
SET CPUBT <CPU-Nummer>
```

Die Nummerierung beginnt mit eins.

1.4.5.20 SET CPUTCT

Dieser Befehl gibt an, auf welchem CPU-Kern der Cyclic-Task laufen soll.

Syntax

```
SET CPUTCT <CPU-Nummer>
```

Die Nummerierung beginnt mit eins.

1.4.5.21 SET CPURT

Dieser Befehl gibt an, auf welchem CPU-Kern der Realtime-Task laufen soll.

Syntax

```
SET CPURT <CPU-Nummer>
```

Die Nummerierung beginnt mit eins.

1.4.5.22 SET DATE

Ändert die Datumseinstellung.

Syntax

SET DATE

Dieser Befehl hat keine Parameter. Sie werden aufgefordert, in das System gültige Werte für das Jahr (von 1990 bis 2199), Monat, Tag sowie Tag der Woche (0 ... Sonntag, 1 Montag, ...) einzugeben.

1.4.5.23 SET DBGLEVEL

Zeigt oder setzt die Konfiguration von Debug-Meldungen.

Syntax

SET DBGLEVEL

Zeigt die Konfiguration von Debug-Meldungen.

SET DBGLEVEL source level

Setzt die konfigurierten Debug-Meldungen.

Betriebssystem RTK

Quelle

Name	Beschreibung	Default-Level
CANLL	Niedrige CAN-Routinen	0
APPHEAP	Applikation-Heap	0
GRAPHIC	Grafikbibliothek	0
SYSHANDLER	Exception-handle	0
CAN	CAN-Kommunikation	0
CDIAS	CDIAS-Bibliothek	0
TASKLISTS	Task-Liste	0

BREAKPOINTINTS	Breakpoint-Handling	0
FLASH	CIPC Flash-Modul	0
IP	TCP/IP communications	0
TGRAPH	LARS-Grafiken	0
SERIAL	Serielle Schnittstellebibliothek	0
USB	USB-Schnittstellenbibliothek (Standard-Level = 2)	2
DEBUGIP	Loader: Debug Interpreter	0
MODLOAD	Modul-Loader	0
KERNEL	LasalOS Kernel (Maintask, Serviceprovider)	0
LINKER	Linker-Bibliothek	0
LOADER	Loader-Bibliothek	0
TASKS	Task Handling	1
VNCCLI	VNC client	1
VNCSVR	VNC server	1
LSLFILE	LslFile-Bibliothek	0
LSLCMD	LslCmd-Bibliothek	0
WEBSVR	WEBSVR-Bibliothek	0
FILESYS	FILESYS-Bibliothek	0
DHCPCLI	DHCPCLI-Bibliothek	0
FTPSERVER	FTPSERVER-Bibliothek	1
LRM	LRM-Bibliothek (Lasal Remote Manager)	0
VARAN	VARAN-Bibliothek	1
VARCAN	VARCAN-Bibliothek (CAN-Geräte)	0
VARSER	VARSER-Bibliothek (serielle Geräte)	0
OSZI	OSZI library (Data Analyzer)	0
SAFETY	SAFETY-Bibliothek	0
SRAMDISK	SRAMDISK library	0

ETHERCAT	ETHERCAT library	0
HWTREE	Hardwaretree library	0
SDIAS	SDIAS-Bibliothek	0
LSLRPT	LASAL Repeater-Bibliothek	0
TRACE	Debugger trace	0
TCP	TCP/IP-Bibliothek	0

Level

Je höher der Level, umso detaillierter die Meldungen.

- | | |
|---|-----------------------------------------------------|
| 0 | Keine Protokollierung |
| 1 | Fehlerprotokollierung |
| 2 | Zusätzliche Protokollierung von Debug-Informationen |
| 3 | Erweiterte Protokollierung von Debug-Informationen |

Betriebssystem Salamander

Quelle

Name	Beschreibung	default level
TRACE	Debugger trace	0
ECATM	EtherCAT Master	0
ECATM_PIMG		0
ECATM_EVENT	EtherCAT Master events	0
ECATM_ATASK	EtherCAT Master asynchronous task	0
ECATM_MONITOR	Ethercat Monitor	0
SAFETY	SAFETY library	0

HWT	Hardware-tree-library	0
VARAN	VARAN library	1
LOG_OS_FILE	FILE I/O interface	0
CIL-SSR	OSKernel library	0
SYSTEM	System	5
LRTMGR	LRT manager	5
LASAL	Log messages written by application	5
GRAPHIC_MEM	_Grafix library	1
LINKER	Linker library	1
DOF	Download on the fly	1
VNCCLI	VNC client	1
VNCSVR	VNC server	0
FILE	FILESYS library	1
FILE-WRITE-CS	File write error callstack	0
GRAPHIC	OS graphics drawing functions based on Xlib.	1
TOUCHE_VENTS	Touch events.	1
KEYEVEN_TS	Key events	1
XEVENTS	X-server events	1
HARDWA RE-IO		0
IXAGENT	Software agent for IXON cloud. Siehe Fußnote 1	1
VPN	Open-VPN library	1
PDF_PRI_NTER	PDF Printer via libHaru	1
SSL	Open-SSL library	1
IP	IP library	1

TOUCHDEVICE	Touch device	1
SERIAL	Serial interface library	1
WLAN	WLAN library	0

Level

Je höher der Level, umso detaillierter die Meldungen.

0	CRITICAL
1	ERROR
2	WARNING
3	INFO
4	DEBUG
5	NOTICE

IXAGENT

Der Debug-Level muss gesetzt werden, bevor der IXAGENT gestartet wird.

Der Ixagent nutzt Log-Levels von 0-9 (gemäß RFC3164) im Gegensatz zu den möglichen Einstellungen von 0-5. Daraus ergibt sich folgenden Zuordnung:

Debug-Level SIGMATEK	Debug-Level IXAGENT
5	9
4	7
3	6
2	4
1	3
0	2

1.4.5.24 SET DIASERROR

Zeigt oder ändert den Dias-Fehler-Setup.

Syntax

SET DIASERROR

Zeigt die aktuelle Dias-Fehlereinstellung an.

SET DIASERROR ENABLE|TRUE|ON

Ermöglicht den DiasError-Interrupt.

SET DIASERROR DISABLE|FALSE|OFF

Deaktiviert den DiasError-Interrupt.

Beispiel

```
SET DIASERROR ON
```

1.4.5.25 SET DISPLAY

Zeigt oder ändert die Farbkalibrierung. Wird der Befehl ohne zusätzliche Optionen aufgerufen, zeigt er nach dem Aufruf die aktuellen Einstellungen. Werden zusätzlich Optionen angegeben, wird der entsprechende Farbparameter gesetzt/geändert.



Diesen Befehl gibt es nur im Salamander-Betriebssystem!

Syntax

SET DISPLAY

Zeigt die aktuelle Farbkalibrierung an.

SET DISPLAY BRIGHTNESS xx

Setzt die Helligkeit des Bildschirms auf den Wert xx. Es kann ein Wert zwischen 0 (extrem dunkel) und 100 (extrem hell) eingestellt werden. Ist der Wert außerhalb dieses Bereichs, wird er auf 100 gesetzt. Der Defaultwert ist 50.

SET DISPLAY SATURATION xx

Setzt den Kontrast des Bildschirms auf den Wert xx. Es kann ein Wert zwischen 0 (extrem blass) und 100 (extrem gesättigt) eingestellt werden. Ist der Wert außerhalb dieses Bereichs, wird er auf 100 gesetzt. Der Defaultwert ist 50.

SET DISPLAY CONTRAST xx

Setzt den Kontrast des Bildschirms auf den Wert xx. Es kann ein Wert zwischen 0 (extrem kontrastarm) und 100 (extrem kontrastreich) eingestellt werden. Ist der Wert außerhalb dieses Bereichs, wird er auf 100 gesetzt. Der Defaultwert ist 100.

SET DISPLAY HUE xx

Setzt den Farbton/die Farbschattierung des Bildschirms auf den Wert xx. Es kann ein Wert zwischen 0 und 360 eingestellt werden. Ist der Wert außerhalb dieses Bereichs, wird er auf 360 gesetzt. Der Default-Wert ist 0.

Beispiel

SET DISPLAY BRIGHTNESS 75

Die Bildschirmhelligkeit wird auf 75 gesetzt

1.4.5.26 SET DISPLAYRESOLUTION

Ermöglicht eine Veränderung der Systemauflösung bzw. ein Umschalten vom Text- in den Grafik-Modus.

Syntax

SET DISPLAYRESOLUTION x

x - 0 - Text Modus (Keine Grafikausgabe möglich)

1 - 320 x 240 px

2 - 640 x 480 px

3 - 800 x 600 px ⁽¹⁾

4 - 1024 x 768 px ⁽¹⁾

5 - 1280 x 1024 px ⁽²⁾

6 - 800 x 480 px ⁽³⁾

(1) Auflösung ist ab LasalOS 01.02.150 verfügbar

(2) Auflösung ist ab LasalOS 01.03.010 verfügbar

(3) Auflösung ist ab LasalOS 01.02.145 verfügbar

Beispiel

SET DISPLAYRESOLUTION 2

Die CPU arbeitet nun im Grafik-Modus mit einer Auflösung von 640 x 480 Pixel.

Bemerkungen

Eine OS-Versionsnummer ab 01.02.130 ist erforderlich. Bei aktiviertem Grafik-Modus steigt der Speicherbedarf des Betriebssystems (320 x 240 => + 150 kB; 640 x 480 => + 600 kB). Dies ist bei einer Anpassung des OS HEAPs zu berücksichtigen.

1.4.5.27 SET DISPLAYROTATE XX

Die Ausgabe der Anzeige kann um 90°, 180° und 270° (VGA-XGA) im Uhrzeigersinn gedreht werden. Der Befehl muss nur einmal eingestellt werden. Nach der erfolgreichen Ausführung startet das System automatisch neu.

Syntax

```
SET DISPLAYROTATE Grad  
Grad 90° / 180° / 270°
```

Beispiel

```
SET DISPLAYROTATE 90  
Die Anzeige wird um 90° im Uhrzeigersinn gedreht.
```

Anforderungen

Eine OS Versionsnummer ab 01.02.022 ist erforderlich.

1.4.5.28 SET FIRSTUSBDRIVE

Legt den ersten zu verwendenden Laufwerksbuchstaben für ein USB-Laufwerk fest.

Der Befehl ist in der autoexec.lsl einzufügen.

Dieser Befehl steht bei allen Steuerungen mit dem Betriebssystem Salamander zur Verfügung und beim Betriebssystem RTK, wenn es auf einer Steuerung mit X86-kompatibler EDGE-Technologie läuft. Siehe Hardware-Beschreibung der Steuerung.

Beim Betriebssystem RTK kann der Laufwerksbuchstabe im Bereich von E: bis F: eingestellt werden.

Beim Betriebssystem Salamander kann der Laufwerksbuchstabe im Bereich von E: bis Z: eingestellt werden.

Syntax

```
SET FIRSTUSBDRIVE F:
```

Legt den ersten Laufwerksbuchstaben für einen angeschlossenen USB-Stick mit F: fest.

1.4.5.29 SET FORCEXTSINGLE

Konfiguriert den Xtimer fest auf den Counter-Modus SINGLE_SHOT. Dies ist für ältere Applikationen nützlich, die den CONTINUOUS Mode nicht unterstützen.

Wird dieses Flag in der Autoexec.lsl gesetzt, dann kann die Applikation den Timer-Mode nicht nachträglich ändern.



Syntax

```
SET FORCEXTSINGLE 1|0
```

Aktiviert/Deaktiviert den festen SINGLE_SHOT Betrieb des Xtimers.

1.4.5.30 SET IP

Zeigt Infos oder setzt die IP-Parameter

Syntax

```
SET IP
```

Zeigt Informationen über den IP-Adresse.

```
SET IP HOSTADDR ip-address
```

Setzt die Host-IP-Adresse.

```
SET IP SUBNET ip-address
```

Setzt die Subnetzmaske.

Die Subnetzmaske muss vor der Gateway-Adresse gesetzt werden.



```
SET IP GATEWAY ip-address
```

Setzt die Gateway-IP-Adresse.

```
SET IP DNS ip-address
```

```
SET IP DNS ip-address_2
```

Betriebssystem RTK: Setzt die erste und zweite (optional) DNS-Serveradresse. Der Befehl ohne eine IP-Adresse zeigt die Einstellungen an.

```
SET IP DNS ip-address[ ip-address_2]
```

Betriebssystem Salamander: Setzt die erste und zweite (optional) DNS-Serveradresse. Der Befehl ohne eine IP-Adresse zeigt die Einstellungen an.

Erfordert (OS > 01.02.160)

```
SET IP PORT
```

Zeigt den eingestellten Online-Port an.

```
SET IP PORT port-address
```

Ermöglicht ein Setzen des Onlineports (Default = 1954)

```
SET IP SPEED Mbps
```

Ermöglicht ein Setzen einer Datenübertragungsrate (z.B. 100 Mbit/s oder 1000 Mbit/s). In der Datei autoexec.lsl muss die Angabe der Geschwindigkeit nach dem Befehl „SET IP X HOSTADDR ip-address“ erfolgen.

```
SET IP X HOSTADDR ...
```

X: 1 ... zweite Ethernet-Schnittstelle ...

```
SET IP ONLTIMEOUT seconds
```

Legt das Timeout der TCP/IP-Online Verbindung fest. Wenn es keine Aktivität bei der Online-Verbindung für mehr als der angegebene Timeout-Wert gibt, dann wird die Verbindung geschlossen.

```
SET IP IRQHOLDOFF x
```

x 1 ... 7 Zyklen auf CyclicTask-Priorität

Ethernet-Interrupts können über mehrere Perioden gesperrt werden, um einen Runtime-Error zu verhindern. Es wird die Anzahl der CyclicTask-Zyklen, für die der IP-Task/Interrupt gesperrt ist, übergeben. Dadurch wird die Systembelastung auf Grund eintreffender TCP/IP-Pakete reduziert.

Dies hat mitunter negative Auswirkungen auf die TCP/IP-Kommunikation, da diese dadurch langsamer wird (LasalOS 01.01.248).



Beispiel

```
SET IP HOSTADDR 198 166 96 204
SET IP SUBNET 255 255 255 0
SET IP GATEWAY 255 255 255 255
SET IP DNS 198.166.96.1
SET IP PORT 1955
SET IP IRQHOLDOFF 2
SET IP 1 HOSTADDR 192.166.96.204 SUBNET 255.255.255.0
SET IP 2 HOSTADDR 192 166 100 204 SUBNET 255 255 255 0
```

Bemerkungen

Bevor diese Werte geändert werden können, muss der TCP-Server mit dem Befehl IP-SHUTDOWN heruntergefahren werden. Nach der Änderung kann der Server mit IP-RESTART neu gestartet werden.

1.4.5.31 SET IP PORT

Diese Methode kann verwendet werden, um den Port für eine bestimmte Ethernet-Schnittstelle einzustellen oder zu ändern.



Port 1000 ist reserviert für den Base Server
Port 1955 ist reserviert für den Comlink Command Server
Port 1956 ist reserviert für den Comlink Refresh Server
Port 1957 ist reserviert für den Alarm LSE Server
Diese Ports dürfen nicht verwendet werden!

1.4.5.32 SET KEYS

Zeigt oder ändert die aktuelle Tastatureinstellung.



Diese Einstellung kann jederzeit geändert werden, beim Drücken von STRG-ALT-F1 für die USA und STRG-ALT-F2 für das deutsche Tastaturlayout.

Syntax

SET KEYS

Zeigt der aktuelle Tastatureinstellung.

SET KEYS US

Setzt US-Tastatur-Layout.

SET KEYS GERMAN

Stellt deutsches Tastatur-Layout (Standard) ein.

SET KEYS NO_LANG_HOTKEYS

Die Kombination „STRG(links)-ALT-F?“ zum Wechseln der Sprache der Tastatur wird nicht abgefangen.

Beispiel

SET KEYS GERMAN

1.4.5.33 SET LASALCOM

Wählt den COM-Port (RS232), der zur Online-Kommunikation verwendet wird, und setzt die Parameter der ausgewählten Schnittstelle.

Syntax

SET LASALCOM NONE

Deaktiviert die Online-Kommunikation mit dem RS232.

SET LASALCOM port BAUD baudrate

Aktiviert Online-Kommunikation neu

port - COM-Interface: COM1 | COM2

baudrate - 9600

14400

19200

38400

56000

115200

Beispiel

SET LASALCOM COM1 BAUD 56000

Bemerkungen

Auf Plattform IPC und CIPC wird die LASALCOM Service auf COM1 gestartet, auch wenn sie nicht in autoexec.lsl gesetzt ist! Um COM1 für ein anderes Gerät (z.B. für ein Touch-Panel) verwenden zu können, leiten Sie LASALCOM zu einem anderen COM weiter (z.B. SET LASALCOM COM2) oder deaktivieren Sie den Service (SET LASALCOM NONE). Wird dies nicht gemacht, wird das Gerät an COM1 nicht korrekt funktionieren.

1.4.5.34 SET LCD

Zeigt oder ändert die aktuellen LCD-Anzeigeeinstellungen.

Dieser Befehl wird nicht auf allen Plattformen unterstützt.



Syntax

SET LCD

Zeigt die aktuellen LCD-Einstellungen.

SET LCD BRIGHTNESS value

Setzt die LCD-Helligkeit (0-255).

SET LCD CONTRAST value

Setzt den LCD-Kontrast (0-255).

Beispiel

SET LCD BRIGHTNESS 128

SET LCD CONTRAST 128

1.4.5.35 SET MAINTIMER

Zeigt oder setzt die MainTimeBasis.

Syntax

SET MAINTIMER

Zeigt das Zeitfenster von MainTimeBasis.

SET MAINTIMER value

Hier können Sie das Zeitfenster für MainTimebasis setzen.

Wert - Gültige Laufzeiten in Mikrosekunden für das Betriebssystem RTK sind: 50, 100, 125, 200, 250, 500, 1000 oder 2000

Anforderungen

CIPCs benötigen eine Xilinx Version ab 2.1

1.4.5.36 SET MOUSE

Zeigt oder setzt Maus/Touch-Typ und die Parameter.

Syntax

SET MOUSE

Zeigt die aktuellen Einstellungen der Maus.

SET MOUSE [TYPE mousetype] [parameter]

Setzt der Maus und die Parameter.

Maus-Typ

KEINE	Keine Maus
PS/2	Maus an der PS/2 Schnittstelle
COM1	Maus an der COM1-Schnittstelle
COM2	Maus an der COM2-Schnittstelle
OM3	Maus an der COM3-Schnittstelle
USB	Maus an der USB-Schnittstelle
TOUCH TERMINAL	Touch (alte Hardware) auf COM3 - siehe unten
TOUCHPS/2	Touch auf PS/2 Schnittstelle
HAMPSHIRE	Touch (alte Hardware) auf COM3 - siehe unten
HAMCOM2	Touch (alte Hardware) auf COM2 - siehe unten
HAMUSB	Touch (neue Hardware) on USB - siehe unten

Parameter

PORt adr	Grundadresse des Ports (Hexadezimal-Format)
IRQ irqnum	Port Interrupt-Nummer
BAUD baudrate	Port Baudrate (nur für Touch!)
CURSOR style	Cursorstil (0-6)



Bitte wenden Sie sich an Ihre Hardware-Dokumentation um zu erfahren, ob es sich um eine alte Hardware-Version (ohne Hampshire) oder eine neue Hardware-Version (mit Hampshire) handelt.



Im Hinblick auf die Verwendung der virtuellen Touch-Tastatur (z.B. in einer C-IPC CLI), muss ein gültiges Touch-Interface konfiguriert werden, sowie die *.SML-Dateien im Verzeichnis C:\lslsys platziert werden. Diese Dateien sind auf der Master-CF-Karte zu finden.

Beispiel

```
SET MOUSE TYPE TOUCHPS/2
SET MOUSE TYPE COM3 PORT 0x3E8 IRQ 10 CURSOR 2
```

Für USB-Geräte und alle Touch-Geräte mit Ausnahme von HAMPSHIRE werden die Parameter PORT, IRQ und BAUD ignoriert. HAMPSHIRE ist standardmäßig auf COM3, diese Einstellungen können aufgehoben werden.

Anforderungen

LasalOS: alle USB-bezogenen Geräte erfordern LasalOS ab Version 5.42.

1.4.5.37 SET MULTICOREOBJS

Zeigt oder setzt die Anzahl der zusätzlichen Prozessorkerne, die eigene Echtzeit- und zyklische Tasks ausführen.

Auf einem, in der Regel dem ersten Prozessorkern einer CPU, laufen immer ein Echtzeit- und ein zyklischer Task. Auf Multi-Core-CPUs unterstützen Salamander und Gecko OS die Ausführung mehrerer Echtzeit- und zyklischer Tasks. Bei Verwendung in der Datei AUTOEXEC.LSL legt der Befehl 'SET MULTICOREOBJS <num_cores>' die Anzahl der zusätzlichen Prozessorkerne fest, die ihre eigenen Echtzeit- und zyklischen Aufgaben ausführen. Mit diesem Befehl kann nicht festgestellt werden, welche Kerne diese Aufgaben ausführen. Der Befehl kann auch über (remote) CLI verwendet werden, um die aktuelle Anzahl der zusätzlichen Prozessorkerne zu ermitteln, die ihre eigenen Echtzeit- und zyklischen Tasks ausführen.

Syntax

```
SET MULTICOREOBJS [<num_cores>]
```

Das Argument <num_cores> muss eine ganze Zahl zwischen null und der Anzahl der dem Betriebssystem zur Verfügung stehenden Prozessorkerne minus eins sein; Verwendung ist nur in AUTOEXEC.LSL erlaubt, wo sie obligatorisch ist; die Gesamtzahl der auf dem System laufenden Echtzeit- und zyklischen Aufgaben wird jeweils als <num_cores> + eins angegeben.

Unabhängig von der Anzahl der dem Betriebssystem zur Verfügung stehenden Prozessorkerne dürfen nicht mehr als sieben zusätzliche Kerne eigene Echtzeit- und zyklische Tasks ausführen, so dass insgesamt höchstens acht Echtzeit- und zyklische Tasks auf der CPU ausgeführt werden können.

Bei Verwendung (ohne Argument) auf (remote) CLI-Befehl wird die aktuelle Einstellung zurückgegeben.

1.4.5.38 SET MULTI_VM

Dieser Befehl ist nur mit dem Salamander-Betriebssystem verfügbar. Bei Verwendung in der Datei AUTOEXEC.LSL aktiviert bzw. deaktiviert SET MULTI_VM [ON|1|OFF|0] die Betriebssystemunterstützung für mehrere VARAN-Manager (VMs), sofern mehrere VMs in der Hardware verfügbar sind. Der Befehl kann auch über (remote) CLI verwendet werden, um den aktuellen Stand der Unterstützung für mehrere VMs und die Anzahl der vom Betriebssystem initialisierten VMs zu ermitteln.

Syntax

```
SET MULTI_VM [ON|1|OFF|0]
```

Das Argument ist nur in AUTOEXEC.LSL zulässig, wo es obligatorisch ist..

Bei Verwendung (ohne Argument) auf (remote) CLI-Befehl wird die aktuelle Einstellung zurückgegeben.

Anforderungen

Salamander OS V09.02.050 oder höher auf x86-SPSen (Verwendung auf (Remote-)CLI wird ab Salamander OS V09.07.112 unterstützt)

Salamander OS V09.03.120 oder höher auf ARM-SPSen (Verwendung auf (Remote-)CLI wird ab Salamander OS V09.04.080 unterstützt)

1.4.5.39 SET OSHEAP

Zeigt oder setzt die Größe der OSHeap.

Syntax

SET OSHEAP

Zeigt die OSHeap Informationen.

SET OSHEAP heapsize

Legt die Eigenschaften von OSHeap fest

heapsize - Länge in kBytes

SET OSHEAP DEFAULT

Stellt den ursprünglichen Wert ein

Anforderungen

LasalOS: LasalOS ab 01.01.220 wird benötigt.

1.4.5.40 SET OSZIMEM

Zeigt oder setzt die Größe des Speichers für den DataAnalyzer.

RTK die Voreinstellung der Speichergröße ist 512 kB. Der Speicher kann im Bereich von 512 kB bis zur Größe von APPDATA eingestellt werden.

Salamander die Voreinstellung der Speichergröße ist 100 kB. Der Speicher kann im Bereich von 100 kB bis 4194303 kB eingestellt werden.

Syntax

SET OSZIMEM

Zeigt die Größe des Speichers für den DataAnalyzer in kByte.

SET OSZIMEM Wert

Setzt die Größe des Speichers für den DataAnalyzer in kBytes.

Wert - Länge \geq 512 kByte bis APPDATA (Betriebssystem RTK)

Wert - Länge \geq 100 kByte bis 4194303 kB (Betriebssystem Salamander)

Bemerkungen

Das Betriebssystem RTK legt den Speicher im Applikationsdaten-Speicherbereich APPDATA an.

Anforderungen

LasalOS: ab 01.02.45 wird benötigt.

1.4.5.41 SET PRINTER

Aktiviert den Drucker und meldet die Drucker-Schnittstelle an.

Syntax

SET PRINTER

1.4.5.42 SET RAM

Zeigt oder setzt die Parameter zur Verarbeitung der RAM-Backup-Datei (SRAM.DAT/ACTIVE.DAT).

Syntax

SET RAM

SET RAM DISPLAY

Zeigt die aktuellen Einstellungen für die RAM-Backup-Datei Verarbeitung.

SET RAM LOG ON|OFF

Schaltet die Protokollierung der Verarbeitung der RAM Backup-Datei ein bzw. aus. Ist der Protokollierung aktiv, werden die Nachrichten im C:\SRAM.LOG oder C:\ACTIVE.LOG geschrieben. Der Standardwert ist OFF.

SET RAM RESET ON|OFF

Ist RAM RESET auf ON eingestellt, werden die gepufferten Daten beim Zurücksetzen der Applikation gesichert. Der Standardwert ist ON.

SET RAM POWERDOWN ON|OFF

Ist RAM POWERDOWN auf ON eingestellt, werden die gepufferten Daten bei der Erkennung einer Powerdown geschrieben. Der Standardwert ist ON.

SET RAM FORMAT OLD|NEW

Legt das Format der RAM-gepufferten Daten der Backup-Datei fest. Der Standardwert ist OLD. Es wird sehr empfohlen, den Wert beim Erstellen der Applikation mit einer Version ab 0.60 auf NEW einzustellen.

1.4.5.43 SET ROUTINGTABLE

Setzt RoutingTable von einem bestimmten Dateinamen.

Syntax

```
SET ROUTINGTABLE
```

Die Routing-Tabelle wird gelöscht

```
SET ROUTINGTABLE Wert
```

Setzt RoutingTable von einem bestimmten Dateinamen.

Wert - <filename.ext>

1.4.5.44 SET RTRUNTIME

Zeigt oder setzt den Runtime-Zähler.

Syntax

```
SET RTRUNTIME
```

Zeigt den Realtime Runtime-Zähler.

```
SET RTRUNTIME Wert
```

Setzt den Realtime Runtime-Zähler.

Wert – die Anzahl der Maintimer Ticks, Bereich: 0-255, 0 = DISABLED

Die Zeit pro Maintimer-Tick kann mit dem CLI-Befehl [SET MAINTIMER](#) eingestellt werden.

1.4.5.45 SET RUNTIME

Displays or sets the runtime counter.

Ist der Runtime Zähler aktiviert, kontrolliert ein Watchdog die Ausführungszeit eines zyklischen Tasks (cywork Methode). Wenn die Ausführungszeit eines zyklischen Tasks den von dem Runtime-Zähler bestimmten Grenzwert überschreitet, stoppt die Anwendung mit dem Status RUNTIME.

Syntax

```
SET RUNTIME
```

Zeigt den Runtime-Zähler.

```
SET RUNTIME value
```

Setzt den Runtime-Zähler.

value – die Anzahl der Ticks (1 Tick = 10 ms), Bereich: 0..255, 0 = DISABLED

1.4.5.46 SET SECURE

Zeigt Informationen oder Einstelloptionen für den sicheren Online-Kanal an.

Syntax

SET SECURE

Zeigt Informationen über den sicheren Online-Kanal an.

SET SECURE VERIFYCLI on/off

Aktivieren oder Deaktivieren der Client-Zertifikatsverifizierung.

SET SECURE CERTPATH path

Legt den Pfad für die Zertifikatsspeicherung fest.

SET SECURE PORT port

Ermöglicht die Einstellung des sicheren Online-Ports (Default = 2054).

1.4.5.47 SET SRAMRETAIN

Zeigt oder setzt die Größe des zugeteilten SRAM-Speichers für globale Variablen.

Syntax

SET SRAMRETAIN

Zeigt die Größe des zugeteilten SRAM-Speichers für globale Variablen.

SET SRAMRETAIN Wert

Setzt den SRAM-Speicher für globale Variablen.

Wert -Länge in Bytes

Anforderungen

LasalOS: ab 01.01.161 wird benötigt.

1.4.5.48 SET STATION

Zeigt oder setzt die Station-ID einer CPU.

Die Kennung muss vor der Routing-Tabelle gesetzt werden!



Syntax

SET STATION

Zeigt die Station-ID einer CPU.

SET STATION value

Setzt die Station-ID einer CPU.

Wert – Bereich: 0-254

1.4.5.49 SET STOPWAIT_TIME

Setzt die maximale Zeit, wie lange das Betriebssystem bei einem [REBOOT](#)-Befehl aus der Applikation wartet, bis die retentiven Server File und Ramex File-Daten ins File geschrieben haben, bevor der [REBOOT](#)-Befehl ausgeführt wird.

Syntax

SET STOPWAIT_TIME Wert

Setzt die maximale Wartezeit in Millisekunden. Der Wert 0 bedeutet, dass die Wartefunktion nicht aktiv ist.

Anforderungen

Die Funktion zum Warten bis alle retentiven Server File und Ramex File-Daten geschrieben haben, ist ab folgenden Versionen verfügbar:

LasalOS ab 01.03.110

Loader ab 02.02.186

Ramex ab 1.17

1.4.5.50 SET TIME

Ändert die aktuelle Zeiteinstellung.

Syntax

SET TIME

Dieser Befehl hat keine Parameter. Sie werden durch das System aufgefordert, um gültige Werte für die Stunde, Minute und Sekunde einzugeben.

1.4.5.51 SET VISU

Legt die Priorität des Visualisierungs-Tasks fest.

Syntax

```
SET VISU HIGH|LOW
```

Die Visualisierung läuft im Background-Task. Mit diesem Befehl können Sie die Priorität der Aufgabe auf eine höhere Priorität setzen. Die Standardeinstellung ist LOW.

1.4.5.52 SET WATCHDOG

Konfiguriert den Watchdog.

Syntax

```
SET WATCHDOG ON|OFF
```

Aktiviert/deaktiviert den Watchdog.

```
SET WATCHDOG Intervall
```

Stellt Watchdog-Intervall in ms ein, $0 < \text{Intervall} \leq 255$

1.4.5.53 TOUCHCFG

Lädt die Touch-Kalibrierungswerte aus einer C:\TOUCHDAT.CFG Datei.



Im Hinblick auf die Verwendung der virtuellen Touch-Tastatur (z.B. in einer C-IPC CLI), muss ein gültiges Touch-Interface konfiguriert werden, sowie die *.SML-Dateien im Verzeichnis C:\lslsys platziert werden. Diese Dateien sind auf der Master-CF-Karte zu finden.

Syntax

```
TOUCHCFG
```

1.4.5.54 TOUCHTEST

Lädt die Touch-Kalibrierungswerte aus einer C:\TOUCHDAT.CFG Datei und startet einen einfachen Touch-Test.

Syntax

```
TOUCHTEST
```

1.4.5.55 UPDATETOUCH

Aktualisiert den Touch-Controller

- Netix im RTK: Für die Aktualisierung der Konfiguration eines Multitouches von Netix müssen als Parameter 'NETIXCFG' und der Pfad zum Konfigurationsfile angegeben werden.
Das Konfigurationsfile muss im OBP-Format (RAW) vorliegen.
- EGalaxy im Salamander: für das Update muss nur der Pfad des Verzeichnisses angegeben werden. Auf dem Speicherplatz müssen das Update-Tool (eUpdate2_ARM), das Firmware-File (fw.bin) und das Konfigurationsfile (SensorTestDefault.ini) liegen.

Syntax

```
UPDATETOUCH NETIXCFG c:\config.raw
UPDATETOUCH f:\egalaxy\
```

1.4.5.56 US

Setzt der Anzeige des Datums bzw. der Uhrzeit auf das US-Format.

Syntax

US

Um wieder auf das europäische Format zu schalten, verwenden Sie den Befehl [EURO](#).

1.4.6 Batch-Verarbeitungsbefehle

1.4.6.1 CLS

Löscht den Bildschirm.

Syntax

CLS

1.4.6.2 DELAY

Verzögert um die angegebene Zeit.

Syntax

DELAY time

Verzögerungszeit in Sekunden

1.4.6.3 ECHO/@ECHO

Ausgabe des Texts nach dem Stichwort auf der Konsole.

@ECHO unterdrückt die Ausgabe der Quellzeile der Batch-Datei und gibt nur den Text aus.

Syntax

```
ECHO ... text ...
@ECHO ... text ...
```

1.4.6.4 EXIT

Stoppt die Ausführung der Batch-Datei.

Syntax

```
EXIT
```

1.4.6.5 GOTO

Führt die Batch-Datei beim Label "label" weiter aus, dem ein ":" vorangestellt sein muss.

Syntax

```
GOTO label
```

Beispiel

```
@ECHO this is a test
goto label1
@ECHO we won't arrive here

:label2
@ECHO second label
goto ende

:label1
@ECHO first label
goto label2

:ende
```

1.4.6.6 IF EXISTS

Bedingte Ausführung der CLI-Befehle, abhängig von der Existenz einer Datei.

Syntax

IF [NOT] EXISTS filename cli-command

filename: muss im 8:3-Format sein

cli-command: jeder gültige CLI-Befehl

1.4.6.7 PAUSE

Unterbricht die Verarbeitung einer Batch-Programm und zeigt eine Meldung an, die den Benutzer dazu auffordert, eine beliebige Taste zum Fortfahren zu drücken.

Syntax

PAUSE

1.4.6.8 REM

Ermöglicht Ihnen Kommentare in einer Batch-Datei oder in Ihrem AUTOEXEC.LSL Datei einzufügen. Der REM-Befehl ist ebenfalls hilfreich zur Deaktivierung von Kommandos.

Syntax

REM [string]

Beispiel

```
rem SET MOUSE TYPE TOUCH
Old cmd line.
rem Now I don't use the touch anymore-Kommentar.
Comment
SET MOUSE TYPE NONE
New cmd line.
```

1.4.7 Umgebungsvariablen

1.4.7.1 FILENAMES

Legt fest, wie Dateinamen mit dem CLI-Befehl DIR angezeigt werden.

Syntax

SETENV FILENAMES SHORT

Dies ist die Standardeinstellung. Die Dateien werden mit ihrem kurzen 8.3-Namen angezeigt.

SETENV FILENAMES LONG

Die Dateien werden mit ihrem vollen Namen (bis zu 256 Zeichen) gezeigt.

1.4.7.2 REBOOTONPF

Legt fest, ob die SPS wird nach einem Stromausfall neu startet.

Jeder Stromausfall wird im System-Log geschrieben.

Syntax

SETENV REBOOTONPF 1

Dies ist die Standardeinstellung. Bei einem Stromausfall wird der Applikation sofort gestoppt, dann wartet die SPS, bis der Strom wiederhergestellt ist und startet neu.

SETENV REBOOTONPF 0

Keine besondere Stromausfall-Aktion.

1.4.8 LasalOS Dateifehler Rückgabe-Codes

Nummer	Fehler-Code	Beschreibung
0	NO_ERROR	Kein Fehler. Dieser Wert zeigt den Erfolg einer Operation an.
-1	ERROR_RESERVED	Dieser Fehler-Code ist reserviert und wird nicht von der FileI/O Klasse zurückgegeben.
-2	PARAM_ERROR	Die Parameter, die an eine FileI/O Klasse-Funktion übergeben werden, sind ungültig. Zum Beispiel die Flags, die an RTFOpen werden, sind widersprüchlich oder die Größe eines String-Puffers ist zu klein.
-3	INVALID_FILENAME	Ein Gerät, Verzeichnis oder die Dateinamen, die an die FileI/O Klasse übergeben wurde, haben eine ungültige Syntax, enthalten unzulässige Zeichen oder MAX_PATH (80) Zeichen überschreitet.

-4	DRIVE_NOT_FOUND	Das Programm versuchte auf ein nicht vorhandenes logisches Laufwerk zugreifen.
-5	TOO_MANY_FILES	Das Programm versuchte mehr Dateien zu öffnen als es verfügbaren Slots in der Datei Tabelle gäbe.
-6	NO_MORE_FILES	Dieser Wert wird vom RTFFindFirst und RTFFindNext zurückgegeben, wenn keine weiteren Dateien den Suchkriterien entsprechen.
-7	WRONG_MEDIA	Eine Diskette wurde in einem Diskettenlaufwerk ersetzt und die Seriennummer der neuen Festplatte stimmt nicht mit der Seriennummer des originalen Datenträgers überein. Zur Behebung dieses Fehlers muss die originale Diskette eingelegt werden oder muss der Betrieb abgebrochen werden.
-8	INVALID_FILE_SYSTEM	Die Informationen vom Boot-Record eines logischen Laufwerks sind inkonsistent und wahrscheinlich beschädigt. Das Laufwerk kann nicht eingebaut werden.
-9	FILE_NOT_FOUND	Die angeforderte Datei wurde auf der Festplatte nicht gefunden.
-10	INVALID_FILE_HANDLE	Ein Datei-Handle, der an eine der FileI/O Klasse API-Funktionen übergeben wurde, ist ungültig. Entweder er wurde bereits geschlossen oder er wurde nicht durch einen erfolgreichen Aufruf an RTFOpen oder RTFFindFirst zurückgegeben, oder er wurde durch die Entfernung der Medien Hosting-Datei automatisch geschlossen.
-11	UNSUPPORTED_DEVICE	Ein Gerät in der Geräteliste hat ein Gerät angegeben, das anders als DEVICE_DISKETTE oder DEVICE_FDISK im Bereich DeviceType ist.
-12	UNSUPPORTED_DRIVER_FUNCTION	Dieser Fehler wird von Gerätetreiber oder Systemtreiber zurückgegeben. Ein Gerätetreiber zum Beispiel für Read-Only-Geräte würde dieser Fehler zurückliefern, beim Versuch auf das Gerät zu schreiben.
-13	CORRUPTED_PARTITION_TABLE	Die FileI/O Klasse hat inkonsistente Werte in der Partitionstabelle eines Geräts gefunden. Das Gerät kann nicht eingebaut werden.
-14	TOO_MANY_DRIVES	Die Zahl der logischen Laufwerke auf allen Geräten, die in der Geräteliste angegeben sind, überschreitet die interne Laufwerk-Tabelle der FileI/O Klasse.
-15	INVALID_FILE_POS	Ein Aufruf von RTFSSeek versucht den Dateizeiger vor dem Beginn der Datei zu positionieren.
-16	ACCESS_DENIED	Dieser Fehler wird beim Fehler eines Sicherheitschecks zurückgegeben. Ein paar von diesen Checks sind: <ul style="list-style-type: none"> Der Versuch eine Datei zu lesen oder schreiben, aber das Schreibschutz-Attribut ist gesetzt. Der Versuch eine bereits geöffnete Datei zu öffnen und mindestens eine der Dateiinstanzen erfordert einen Schreibzugriff, und SHARED ist nicht für alle Instanzen der Datei angegeben. Versuch, eine Datei zu erstellen und die angegebene Datei mit dem eingestellten Read-Only-Attribut bereits existiert. Der Versuch, ein Volumenlabel zu öffnen.

		<ul style="list-style-type: none"> Der Versuch ein Verzeichnis mit Lese- und Schreibzugriff zu öffnen. Versuch ein Verzeichnis mit Flag OPEN_NO_DIR zu öffnen. Der Versuch, eine Datei mit eingestelltem Schreibschutz, Volumenlabel, oder Verzeichnis zu löschen. Der Versuch einer Datei über Laufwerke neu zu benennen. Der Versuch ein Volumen Label zu umbenennen. Der Versuch ein Verzeichnis nach einem seiner Unterverzeichnisse neu zu benennen. Der Versuch das aktuelle Verzeichnis neu zu benennen. Der Versuch die Attribute, Datum und Uhrzeit, oder die Dateigröße eines Hauptverzeichnisses einzustellen. Der Versuch ein Volumenlabel oder Verzeichnisattribute zu ändern. Der Versuch eine Datei, die nur für Lesezugriff geöffnet ist, zu schreiben, kürzen oder erweitern. Versuch ein Verzeichnis zu entfernen, das nicht leer ist. Der Versuch ein Verzeichnis mit eingestelltem read-only Attribut zu entfernen. Der Versuch das Haupt- oder aktuelle Verzeichnis zu entfernen. ResetDisk wurde aufgerufen, während Dateien auf dem Ziel-Drive geöffnet sind.
-17	STRING_BUFFER_TOO_SMALL	Die Größe eines String-Puffers, der an einen der FileI/O-Klasse übergeben wurde, ist zu klein, um das Ergebnis zu enthalten.
-18	GENERAL_FAILURE	Ein Gerätetreiber hat einen Fehler gemeldet, ohne zusätzlichen Informationen über die Ursache zu liefern. Zum Beispiel, nicht vorhandene Hardware oder tödliche Hardwareausfälle könnten diesen Fehler erzeugen.
-19	PATH_NOT_FOUND	Der Pfad für eine Datei oder ein Verzeichnis, der an den FileI/O-Klasse übergeben wurde, wurde nicht gefunden.
-20	FAT_ALLOC_ERROR	Die FileI/O-Klasse hat ungültige Werte in einer FAT-Partition gefunden. Die FAT ist sehr wahrscheinlich beschädigt und die Partition muss neu formatiert werden.
-21	ROOT_DIR_FULL	Es wurde versucht, eine neue Datei in einem Hauptverzeichnis zu erstellen, das jedoch voll ist. Im Unterschied zu Unterverzeichnissen haben Hauptverzeichnisse eine feste Größe und können nicht vergrößert werden.
-22	DISK_FULL	Es wurde versucht eine Datei oder ein Verzeichnis zu vergrößern, aber es wurden nicht ausreichend Cluster zur Erfüllung der Anfrage gefunden. Für die Funktion RTFExtend, wird dieser Fehler zurückgegeben, wenn nicht ausreichend durchgehende Cluster gefunden werden.
-23	TIMEOUT	Dieses Gerät-Fehler wird gemeldet, wenn ein Gerät nicht innerhalb eines angemessenen Zeitraums reagiert.

-24	BAD_SECTOR	Ein Gerätetreiber hat gemeldet, dass ein Sektor auf der Festplatte nicht gelesen oder geschrieben werden konnte.
-25	DATA_ERROR	Ein Gerätetreiber hat berichtet, dass ein von der Festplatte gelesener Sektor die Datenintegritätsprüfung nicht bestanden hat. Normalerweise werden die Daten mit CRC-Checksumme gespeichert, die zur solchen Kontrollen verwendet wird.
-26	MEDIA_CHANGED	Während eines Gerätezugriffs hat der Treiber festgestellt, dass die Medien des Laufwerks entfernt oder ausgetauscht wurden.
-27	SECTOR_NOT_FOUND	Ein Gerätetreiber konnte den angeforderten Sektor nicht finden. Entweder ein beschädigter Boot-Sektor oder Low-level Formatierung kann zu diesem Fehler führen.
-28	ADDRESS_MARK_NOT_FOUND	Die Adressemarke, die wird normalerweise während einer Low-Level Formatierung geschrieben, wurde während einer Festplatte Lese- oder Schreiboperation nicht gefunden.
-29	DRIVE_NOT_READY	Eine Festplatte Gerät reagiert nicht, weil es entweder nicht vorhanden ist, die Medien nicht eingesteckt ist, oder wegen einen Hardware-Fehler.
-30	WRITE_PROTECT	Es wurde versucht auf ein schreibgeschütztes Medium zu schreiben.
-31	DMA_OVERRUN	Einen DMA-Controller hat diesen Fehler gemeldet. Das kann passieren, wenn ein DMA-Puffer eine 64k-Adresse Grenze enthält.
-32	CRC_ERROR	Eine CRC-Prüfung ist während einer Gerät-Lese- oder Schreiboperation fehlgeschlagen.
-33	DEVICE_RESOURCE_ERROR	Ein Gerätetreiber hat berichtet, dass einige benötigten Ressourcen nicht verfügbar sind. Zum Beispiel, der Floppy-Treiber konnte einen DMA-Puffer nicht allozieren oder der RAM-Disk-Treiber konnte einen Platz für neue Sektoren nicht zuordnen.
-34	INVALID_SECTOR_SIZE	Ein Gerät berichtet mit einer Nicht-Standard Sektorgröße formatiert zu werden. Normalerweise sollten FAT-Medien Sektoren von 512 Bytes-Größe verwenden. Bitte informieren Sie sich, wie die File/O-Klasse andere Sektorgrößen unterstützen kann.
-35	OUT_OF_BUFFERS	Die File/O-Klasse war nicht in der Lage, einen neuen Puffer in seinem internen Puffer-Cache zu allozieren. Diese Situation kann nur auftreten, wenn alle Puffer verschmutzt sind und keiner den schmutzigen Puffern auf dem gleichen physikalischen Gerät sind, wobei ein neuer Puffer erforderlich ist. Um diese Fehler zu vermeiden, erhöhen Sie die Anzahl der Puffer, beenden einige Dateien oder löschen Puffer, bevor die fehlerhafte Funktion aufgerufen wird.
-36	FILE_EXISTS	Dieser Fehler wird zurückgegeben bei einem Versuch eine Datei auf einen vorhandenen Dateinamen umzubenennen oder bei der Erstellung eines Verzeichnisses mit dem Namen einer vorhandenen Datei oder eines vorhandenen Verzeichnisses.

-37	LONG_FILE_POS	Dieser Rückgabe-Code stellt keinen Fehler dar. Er wird von RTFSeek zurückgegeben, wenn der Wert des neuen Dateipointers 231-1 (2G minus eins) überschreitet. Allerdings ist die Funktion erfolgreich, wenn dieser Wert zurückgegeben wird. Verwenden Sie die RTFGetFileInfo Funktion zum Abrufen des aktuellen Dateizeigers.
-38	FILE_TOO_LARGE	Die Anwendung hat versucht eine Datei zu vergrößern, um 4G oder mehr Bytes zu enthalten. Allerdings unterstützen FAT Dateisysteme nur Dateigrößen bis zu FFFFFFFFh Bytes. Diese Einschränkung gilt auch für FAT-32 Partitionen.

1.5 Software-Konfiguration

1.5.1 Software-Komponenten

Das LASAL Betriebssystem besteht aus dem folgenden unterschiedlichen Software-Komponenten.

Operating system	Das Betriebssystem besteht aus einer Image-Datei und der Dateierweiterung .RTB (IPC, IPC-C) oder .BIN (386er-CPUs).
System data	Systemdaten sind z.B. DLLs (Dynamic Loadable Library) oder Steuerdaten für die virtuelle Tastatur des IPC mit Touch, die sich Ordner C:\LSLSYS des Zielsystems befinden.
Batch-Daten und Makros	Befinden sich auf dem Zielsystem im Verzeichnis C:\LSCMD.
LASAL CLASS project	Die LASAL CLASS Projekt-Dateien befinden sich auf dem Zielsystem im Ordner C:\LSLWORK. In LASAL CLASS können die Dateien in diesem Verzeichnis mit dem 'Project / Create Bootdisk' Menübefehl erstellt werden.
Lasal Screen (LSE) project	Die LASAL SCREEN Projekt-Dateien befinden sich auf dem Zielsystem im Ordner C:\MPC. Zur C:\IPC.INI gehört auch das LSE-Projekt. Im PC befinden sich die Dateien des Projekts unter dem Pfad RUNTIME im Unterverzeichnis.
Firmware	Die Firmware für diverse Komponenten (z.B. Xilinx) des Zielsystems können mit einem Software-Befehl upgedatet werden.

1.5.2 Installation der Debug-Tools LASAL Class und LSE

Betriebssystem

Das Betriebssystem kann in LASAL CLASS über den Menüpunkt Debug/Extras/Download Betriebssystem aktualisiert werden. Schalten Sie das Zielsystem während der Aktualisierung des Betriebssystems nicht aus!

System-Dateien, Batch-Dateien und Makros

In LASAL CLASS können diese Dateien im entsprechenden Verzeichnis über den Menüpunkt Debug/Extras/Datei-Transfer zur SPS übertragen werden.

LASAL CLASS-Project

In LASAL CLASS kann ein Projekt in den Programmspeicher über das Menü Zeit Debug/Extras/Datei-Transfer zur SPS übertragen werden. Um das Projekt permanent zu speichern, muss der Menüpunkt Debug/Extras/Save project in the control ausgewählt werden.

LASAL SCREEN-Projekt

In LASAL SCREEN wird das LSE-Projekt mit dem Menüpunkt Debug/Download-Projekt installiert.

Firmware

Die Debug-Tools können nicht zum Aktualisieren der Firmware verwendet werden.

1.5.3 Installation über das Command Line Interface (CLI)

Operating system

Das Betriebssystem kann mit dem CLI-Befehl `BOOT <OS Bild Datei> [<Laufwerksbuchstabe>]` installiert werden. C: ist der Standardwert für <Laufwerksbuchstabe>.

System-Dateien, Batch-Dateien und Makros

Makros, Systemdateien und Batch-Dateien werden installiert, indem Sie die gewünschten Dateien in das entsprechende Verzeichnis kopieren.

LASAL CLASS Project

Ein LASAL CLASS Projekt kann installiert werden, indem Sie die Dateien in den Ordner C:\LSLWORK Verzeichnis kopieren.

LASAL-Screen-Projekt

Ein LSE-Projekt kann installiert werden, indem Sie die Dateien in den Ordner C:\LSLWORK speichern und das Verzeichnis der C:\IPC.INI-Datei kopieren.

Firmware

Ein eigener CLI-Befehl ist für die Aktualisierung der Firmware verfügbar.

Eine Übersicht der verfügbaren CLI-Befehle finden Sie unter [Command Line Interface](#).

1.5.4 Automatische Installation von austauschbaren Speichermedien

Ein austauschbarer Datenträger (Diskette oder USB-Drive) kann über zwei Möglichkeiten installiert werden:

1. Im ausgewählten Medium wird ein AUTOEXEC.LSL mit einem Installationsbefehl gespeichert, das den Laufwerksbuchstaben für den Suchpfad der AUTOEXEC.LSL beinhaltet.

Die Suchfolge nach der AUTOEXEC.LSL-Datei auf dem IPC sieht wie folgt aus:

A:\(Floppy),

D:\(SmartMedia, wenn vorhanden)

C:\(Festplatte)

Die Suchfolge, nach der AUTOEXEC.LSL Datei auf der C-IPC ist:

B:\(USB-Floppy)

E:\(erster USB-Drive mit einer LUN)

D:\(Compact Flash im 2. Steckplatz)

C:\(Compact Flash im 1. Steckplatz)

2. Die AUTOEXEC.LSL am Drive C: enthält Befehle, die zum Starten einer Batch-Datei oder eines Programms von einem austauschbaren Datenträger verwendet werden. Der Pfad für die Rexx-Programme wird zunächst mit dem angegebenen Laufwerksbuchstaben erweitert. Es wird versucht, ein Rexx-Programm zu starten. Schließlich wird der Such-Pfad auf den Standard-Wert zurückgesetzt.

Der Eintrag AUTOEXEC.LSL sieht wie folgt aus:

```
SETENV REXX_MACROS B:\;F:\;G:\;H:\;I:\;J:\;K:\;L:\;M:\;N:\;O:\;  
REXX AUTOSTRT  
SETENV REXX_MACROS C:\LSLCMD
```

Da die Disk-Änderung nicht mehr erkannt werden kann, kann das Rexx-Programm nicht von einer Installations-Disk ausgeführt werden. Der folgende Eintrag ist bei der Verwendung einer Disk daher erforderlich:

```
DEL c:\tmpcmd
COPY a:\autostrt c:\tmpcmd
REXX c:\tmpcmd
DEL c:\tmpcmd
```

1.5.5 Rexx-Programme zur Installation von Software

Es sind Rexx-Programme zur Verwaltung von Software-Paketen verfügbar; ein Software-Paket enthält folgendes:

- Die zu installierenden Dateien und die entsprechende Beschreibungsdatei.
- Verschiedene Software-Komponenten wie ein Betriebssystem und ein LASAL CLASS-Projekt können inkludiert werden.
- Die Möglichkeit auf verschiedene Arten von Speichermedien zuzugreifen (zum Beispiel, wenn die Festplatte voll ist).
- Eine Identifizierungs-Funktion, um Speichermedien zu erkennen, die falsch eingefügt wurden.

Rexx-Anforderungen

- Ab LASALOS 5.42
- Die rexx.dlm-Datei muss im Verzeichnis C:\LSLSYS gespeichert werden.
- Für komprimierten Software-Pakete muss die Datei lslzip.dlm im Verzeichnis C:\LSLSYS gespeichert werden.

Eine detaillierte Beschreibung des Rexx-Interpreters finden Sie in der [Rexx Interpreter](#) Dokumentation.

1.5.6 Beschreibungsdatei eines Software-Pakets

Die Beschreibungsdatei ist eine ASCII-Datei mit dem Namen MEDIUM.INF, die aus einem Header besteht, der das Software-Paket und zusätzliche Einträge zur Beschreibung der einzelnen Dateien auf dem Speichermedium beschreibt.

Der Header ist wie folgt konfiguriert:

```
NAME <Name des Software-Pakets>
DATE <Datum>
DISKNBR <Nummer des Datenmediums, beginnend mit 1>
[ INCOMPLETE ]
```

Die INCOMPLETE-Angabe ist optional und bedeutet, dass weitere Datenmedien zum Software-Paket gehören.

Einträge, die zur Beschreibung der Dateien auf dem Datenträger verwendet werden, sind so aufgebaut:

<Type> <Path> <File> <Command> <Target path> <Group>

<Type>	Markiert den Dateityp.				
	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="background-color: #D9E1F2; width: 15%;">FILE</td><td>Dies bedeutet, dass die Datei entpackt wird.</td></tr> <tr> <td style="background-color: #D9E1F2;">SPLITA RCH</td><td>Dieser Eintrag beschreibt eine Datei oder einen Teil der Datei eines Archivs. Ein Archiv ist eine Datei, in der die Quelldateien in verpackter Form kombiniert sind. PKG sind die Basis-Namen (ohne Dateiendung). Die Datei-Erweiterung wird durch das Archivierungsprogramm definiert, um zum Beispiel die Reihenfolge der Teildateien zu markieren. Das Hauptverzeichnis auf dem Datenträger ist der Pfad zur Archivdatei.</td></tr> </table>	FILE	Dies bedeutet, dass die Datei entpackt wird.	SPLITA RCH	Dieser Eintrag beschreibt eine Datei oder einen Teil der Datei eines Archivs. Ein Archiv ist eine Datei, in der die Quelldateien in verpackter Form kombiniert sind. PKG sind die Basis-Namen (ohne Dateiendung). Die Datei-Erweiterung wird durch das Archivierungsprogramm definiert, um zum Beispiel die Reihenfolge der Teildateien zu markieren. Das Hauptverzeichnis auf dem Datenträger ist der Pfad zur Archivdatei.
FILE	Dies bedeutet, dass die Datei entpackt wird.				
SPLITA RCH	Dieser Eintrag beschreibt eine Datei oder einen Teil der Datei eines Archivs. Ein Archiv ist eine Datei, in der die Quelldateien in verpackter Form kombiniert sind. PKG sind die Basis-Namen (ohne Dateiendung). Die Datei-Erweiterung wird durch das Archivierungsprogramm definiert, um zum Beispiel die Reihenfolge der Teildateien zu markieren. Das Hauptverzeichnis auf dem Datenträger ist der Pfad zur Archivdatei.				
<Path>	Pfad zur Datei auf dem Datenträger				
<File>	<p>Name der Datei auf dem Datenträger (ohne Pfad)</p> <p>Folgende Fehler werden nur beim FILE type interpretiert:</p>				
<Command>	<p>Legt fest, was mit der Datei auf dem Zielsystem passieren soll.</p> <ul style="list-style-type: none"> • COPY • PRJINST • OSINST 				
<Target path>	<p>Einem symbolischen Namen für das Verzeichnis auf dem Zielsystem. Der Effektivwert dieser Variable wird bis zum Zeitpunkt der Installation auf dem Zielsystem nicht festgelegt. Die folgenden Werte sind möglich:</p> <ol style="list-style-type: none"> 1. <Command>=COPY MPC_DIR IPCINI_DIR CMD_DIR LSLSYS_DIR 2. <Command>=PRJINST LSLWORK_DIR 3. Es wird bei der restlichen Werte für <Command> <target path> nicht interpretiert. 				
<Group>	Mit diesem Eintrag können Dateien einem Gruppenname zugewiesen werden. Dabei ist es möglich, dass ein Software-Paket Dateien für mehrere Zielsysteme enthält. Während der Installation kann festgelegt werden, dass nur Dateien von einer speziellen Gruppe installiert werden sollen.				

1.5.6.1 Beispiele für Beschreibungsdateien

1) Ein LASAL CLASS- und ein LSE-Projekt auf dem IPC und das LASAL CLASS-Projekt in einer DCL641:

Die Software-Komponenten des IPCs werden mit dem Namen GRP_IPC und die in der DCL641 mit dem Namen GRP_386 markiert.

```
NAME      PROJECT
DATE      04/01/14,13:57:06
DISKNBR   1

FILE GRP_IPC\LSLWORK index.txt PRJINST LSLWORK_DIR GRP_IPC
FILE GRP_IPC\LSLWORK PROJECT.idx PRJINST LSLWORK_DIR GRP_IPC
FILE GRP_IPC\LSLWORK M0.lob PRJINST LSLWORK_DIR GRP_IPC
FILE GRP_IPC\LSLWORK M1.lob PRJINST LSLWORK_DIR GRP_IPC
.

FILE GRP_IPC\MPC Vis.MPC COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC LT0016.MPC COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC LT0002.MPC COPY MPC_DIR GRP_IPC
.

FILE GRP_IPC\IPCINI IPC.INI COPY IPCINI_DIR GRP_IPC
FILE GRP_386\LSLWORK index.txt PRJINST LSLWORK_DIR GRP_386
FILE GRP_386\LSLWORK M0.lob PRJINST LSLWORK_DIR GRP_386
FILE GRP_386\LSLWORK M1.lob PRJINST LSLWORK_DIR GRP_386
.

FILE GRP_386\LSLWORK PROJECT.idx PRJINST LSLWORK_DIR GRP_386
```

2) Ein Software-Paket im Archiv-Format getrennt auf zwei Festplatten

Disk 1:

```
name      PROJECT
date      04/01/14,13:57:06
disknbr   1

SPLITARCH . PKG.000 NA NA NA
incomplete
```

Disk 2:

```
name      PROJECT
date      04/01/14,13:57:06
disknbr   2

SPLITARCH . PKG.001 NA NA NA
```

1.5.7 Programm zur Erstellung einer Archivdatei

Mit dem ARCH.EXE-Windows-Programm kann der Inhalt eines Verzeichnisses in eine Archivdatei oder in mehrere Teildateien des Archivs kopiert werden. Wenn zu wenig Speicherplatz auf dem Speichermedium vorhanden ist, muss ein Archiv in Teildateien aufgeteilt werden.

Syntax

```
arch <archive name> [parameters]
```

-r<root-dir>	Hauptverzeichnis von <file-spec>, Standard = . Das Verzeichnis enthält die Dateien, die einem Archiv hinzugefügt werden sollen (einschließlich aller Unterverzeichnisse)
-f<file-spec>...	Dateinamen können auch Platzhalter enthalten (* und ?), Std. = *.*
-x	Extraktion von Dateien aus dem Archiv in das Hauptverzeichnis. Ohne diesen Parameter wird ein Archiv erstellt.
-s<splitsize>	Das Archiv ist in Teildateien der <splitsize>-Größe geteilt.
-?	Anzeige der Hilfe.

Beispiel

```
ARCH example.arc
```

C ein Archiv im aktuellen Verzeichnis mit allen Dateien (*.*) aus dem aktuellen Verzeichnis

```
ARCH C:\example.arc -rE:\Temp -f*.txt
```

Erstellt ein Archiv, in C:\ von allen .TXT-Dateien in E:\Temp

```
ARCH example.arc -x
```

Sucht im aktuellen Verzeichnis nach der Archivdatei "example.arc" und extrahiert die Dateien ins aktuelle Verzeichnis

```
ARCH C:\example.arc -red:\Temp -x
```

Extrahiert die Dateien vom Archiv (wenn verfügbar) nach E:\Temp

1.5.8 AUTOSTRT REXX-Programm für die Installation des Software-Packages

Mit dem Programm AUTOSTRT Rexx kann ein Softwarepaket installiert werden. Die Installationsparameter wie die Adresse des Zielsystems und die Zielverzeichnisse werden ebenfalls im ersten Teil des Programms definiert, daher ist es notwendig, die AUTOSTRT-Datei zu bearbeiten und die Parameter anzupassen. Der zweite Teil enthält den Programmcode. Dieser Teil muss nur geändert werden, wenn eine andere Programmkonfiguration gewünscht ist.

Nach der Erstellung des Softwarepaket wird die AUTOSTRT-Datei entweder auf den ersten Datenträger des Softwarepaket oder auf einen separaten Datenträger kopiert, wenn zu wenig Platz vorhanden ist.

Wenn sich der Datenträger mit der AUTOSTRT-Datei beim Hochfahren im Laufwerk befindet und die AUTOEXEC.LSL einen AUTOSTRT-Aufruf enthält, wird die Installation automatisch gestartet.

1.5.8.1 Beispiel

1.5.8.1.1 System Update IPC

Auf dem Wechseldatenträger befinden sich eine AUTOEXEC.LSL und die Betriebssystemkarte (.RTB-Datei).

Der AUTOEXEC.LSL Inhalt sieht so aus:

```
REM +-----+
REM |           LASAL OS Installation
REM |           S i g m a t e k   G m b H & C o K G
REM |           www.sigmatek.at
REM +-----+
REM -----
REM update operating system
REM -----
BOOT a:\lslsys\lasalos.rtb
```

Wenn in der BIOS-Bootreihenfolge das Laufwerk A: vor dem Laufwerk C: definiert ist (BIOS-Eintrag 'Boot Sequence A, C'), erscheint eine Fehlermeldung, die besagt, dass der falsche Datenträger eingelegt ist, weil sich kein bootfähiges Betriebssystem auf der Installationsdiskette befindet. Um diese Fehlermeldung zu vermeiden, ändern Sie die BIOS-Einstellungen oder legen Sie den Datenträger erst nach dem Start des Betriebssystems von C: ein (d.h. nach der Anzeige 'Programm laden ...').

Es wird nicht empfohlen, zu lange zu warten, da die AUTOEXEC.LSL vom Laufwerk C: aus ausgeführt wird.

Nach der Aktualisierung des Betriebssystems ist ein Neustart erforderlich!

1.5.8.1.2 System Update CPU

Es wird ein Softwarepaket erstellt und auf der Installationsdiskette befinden sich die Datei MEDIUM.INF, das Programm AUTOSTRT Rexx und die Betriebssystemkarte (.BIN-Datei). Die CAN-Stationsnummer der abgetrennten CPU ist 2.

Die Datei MEDIUM.INF sieht wie folgt aus:

```
NAME          OS-Update-DCL641-5.42
DATE          04/01/14,13:57:06
DISKNBR       1
```

```
FILE . DCL641.BIN PRJINST NA GRP_DCL
```

Der Parameter in AUTOSTRT:

```
/*
 * Path to the software packet.
 */
settings.pkgsrc_dir      = 'A:\'

/*
 * Directories to temporary files and the log-file
 */
settings.archtmp_dir      = 'C:\TMPINST\ARCH'
settings.pkgtmp_dir        = 'C:\TMPINST\PKG'
settings.logfile           = 'C:\INSTALL.LOG'

/*
 * Timeout waiting for the next installation medium
 */
settings.timeout_sec       = 15

/*
 * The following values are used to resolve variables in MEDIUM.INF.
 * Normally it should not be necessary to change these values.
 */
settings.lslwork_dir       = 'C:\LSLWORK'
settings.mpc_dir            = 'C:\MPC'
settings.ipcini_dir         = 'C:\'
settings.cmd_dir            = 'C:\LSLCMD'
settings.lsldsys_dir        = 'C:\LSSLDSYS'

/*
 * When a value other than 0 is used for useSoftwareKey, then the value from
 * NAME in MEDIUM.INF is compared with the value of NAME in C:\SWKEY.TXT.
 * When the two values do not match, the installation is aborted.
 */
settings.useSoftwareKey     = 0

/*
 * Specifies the number of groups to process
```

```

*/
settings.nbr_of_groups      = 1

/*
 * First group: install to can station 1
 */
settings.0.interface        = 'CAN1'
settings.0.address          = '2'
settings.0.group             = 'GRP_DCL'

```

1.5.8.1.3 LASAL auf mehreren Datenträgern

- 1) Alle zur Installation gehörenden Dateien sind in ein eigenes Verzeichnis zu kopieren, z.B. C:\INST\GRP_IPC.
 - a. LASAL CLASS-Dateien: In LASAL CLASS werden die Dateien mit 'Project / Create bootdisk' im Verzeichnis C:\INST\GRP_IPC\LSLWORK erstellt.
 - b. LASAL-Screen Dateien: Das Verzeichnis <LSE-Project>\RUNTIME wird nach C:\INST\GRP_IPC\RUNTIME kopiert.
 - c. Die Datei <LSE-Projekt>\RUNTIME\IPC.INI wird nach C:\INST\GRP_IPC\IPCINI\IPC.INI kopiert
- 2) Erstellung der Beschreibungsdatei MEDIUM.INF. Inhalt der MEDIUM.INF:

```

name PROJECT
date 04/05/04,15:24:54
disknbr 1

FILE GRP_IPC\LSLWORK index.txt PRJINST LSLWORK_DIR GRP_IPC
FILE GRP_IPC\LSLWORK M0.lob PRJINST LSLWORK_DIR GRP_IPC
FILE GRP_IPC\LSLWORK M1.lob PRJINST LSLWORK_DIR GRP_IPC
FILE GRP_IPC\LSLWORK M10.lob PRJINST LSLWORK_DIR GRP_IPC
.

.

FILE GRP_IPC\IPCINI IPC.INI COPY IPCINI_DIR GRP_IPC
FILE GRP_IPC\MPC Arial_10.fnt COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC Arial_11.fnt COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC Arial_12.fnt COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC Arial_14.fnt COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC Arial_8.fnt COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC Arial_9.fnt COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC ipc.ini COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC LC0000.MPC COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC LC0001.MPC COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC LC0002.MPC COPY MPC_DIR GRP_IPC
FILE GRP_IPC\MPC LC0003.MPC COPY MPC_DIR GRP_IPC
.
.
```

- 3) Da das Softwarepaket auf Festplatten kopiert werden soll, ist die Konvertierung in ein Archivformat praktischer. Bei der Konvertierung in ein Archivformat werden alle zum Softwarepaket gehörenden Dateien in eine Archivdatei gepackt und in kleinere Dateien aufgeteilt, die eine optimale Größe für eine Festplatte haben.

Mit dem Programm ARCH.EXE kann ein komplettes Verzeichnis gepackt und die einzelnen Archivdateien in ein separates Verzeichnis (C:\PKG1AR) kopiert werden.

Programmaufruf:

```
MD C:\PKGAR
ARCH C:\PKGAR\PKG.AR -rC:\INST -f*.* -s1400000
```

Im Verzeichnis C:\PKGAR befinden sich die Dateien PKG.000, PKG001 und PKG.002.

- 4) Die Dateien PKG.000, PKG.001 und PKG.002 werden jeweils auf einen eigenen Datenträger kopiert. Für jeden Datenträger muss eine eigene Datei MEDIUM.INF erstellt werden:

```
Disk 1:
PKG.000
MEDIUM.INF
```

Inhalt von MEDIUM.INF:

```
name PROJECT
date 04/05/04,15:24:54
disknbr 1

SPLITARCH . PKG.000 NA NA NA
incomplete
```

```
Disk 2:
PKG.001
MEDIUM.INF
```

Inhalt von MEDIUM.INF:

```
name PROJECT
date 04/05/04,15:24:54
disknbr 2

SPLITARCH . PKG.001 NA NA NA
incomplete
```

```
Disk 3:
PKG.002
MEDIUM.INF
```

Content of MEDIUM.INF:

name	PROJECT
date	04/05/04,15:24:54

```
disknbr      3
SPLITARCH . PKG.002 NA NA NA
```

- 5) Parameter in der AUTOSTRT-Datei anpassen. Diese Datei wird dann auf die 1. Installationsdiskette kopiert.

Inhalt von AUTOSTRT:

```
/*
 * Path to the software packet.
 */
settings.pkgsrc_dir      = 'A:\'

/*
 * Directories to temporary files and the log-file
 */
settings.archtmp_dir      = 'C:\TMPINST\ARCH'
settings.pkgtmp_dir        = 'C:\TMPINST\PKG'
settings.logfile           = 'C:\INSTALL.LOG'

/*
 * Timeout waiting for the next installation medium
 */
settings.timeout_sec       = 15

/*
 * The following values are used to resolve variables in MEDIUM.INF.
 * Normally it should not be necessary to change these values.
 */
settings.lslwork_dir       = 'C:\LSLWORK'
settings.mpc_dir            = 'C:\MPC'
settings.ipcini_dir         = 'C:\'
settings.cmd_dir             = 'C:\LSLCMD'
settings.lslsys_dir          = 'C:\LSLSYS'

/*
 * When a value other than 0 is used for useSoftwareKey, then the value from
 * NAME in MEDIUM.INF is compared with the value of NAME in C:\SWKEY.TXT.
 * When the two values do not match, the installation is aborted.
 */
settings.useSoftwareKey = 0

/*
 * Specifies the number of groups to process
 */
settings.nbr_of_groups = 1

/*
 * First group: install to can station 1
 */
settings.0.interface = 'LOCAL'
settings.0.address = 'NA'
settings.0.group = 'GRP_IPC'
```

1.6 Rexx Interpreter

1.6.1 Was ist LASALOS Rexx?

LasalOS Rexx ist eine Anwendung der Rexx (Restrukturierte Extended Executor) Sprache, die eine interpretierte Skriptsprache ist und das Schreiben von Programmen in einer strukturierten Form ermöglicht. Der Zweck dieser Applikation ist es, eine Möglichkeit zur Verfügung zu stellen, Vorgehensweisen für administrative Aufgaben zu schreiben, wenn kein LASAL-Projekt wie das Verfahren zur Verteilung von Software und Daten von einer SPS zur anderen läuft. Es ist jedoch nicht vorgesehen, diese Anwendung von einem LASAL-Projekt aufzurufen.

LasalOS Rexx ist in der Lage, beliebige Kommando-Strings zur Ausführung im CLI-(Command Line Interface) Umfeld zu übertragen. Mehrere integrierte LasalOS-spezifische Funktionen sind neben den standardmäßig integrierten Rexx-Funktionen vorhanden.

1.6.2 Der Zweck dieses Dokuments

Dieses Dokument beschreibt die LASALOS Rexx Sprachen-Anwendung und die integrierten LASALOS-spezifischen Funktionen, ist aber keine Referenz zu Rexx. Bitte ziehen Sie andere Dokumentationen für eine Beschreibung der Rexx Sprache heran.

Anforderungen

LasalOS Benötigt LasalOS ab Version V5.42

Platform LASALOS Rexx ist auf IPC und LARS vorhanden.

Einschränkungen

Es werden nur Dateinamen im 8.3-Format unterstützt.

Einbauzustand

Kopieren Sie die Datei IPC Rexx.DLM auf einen IPC in das Zielsystem in das Verzeichnis C:\LSLSYS. Die Datei Rexx.DLM finden Sie im LASAL-Programmverzeichnis (normalerweise C:\Program Files\LASAL) im Unterverzeichnis LASAL\Rexx. In LARS installiert das Lars Setup-Programm automatisch LasalOS Rexx.

Verwendung der LASALOS Rexx

- Wählen Sie ein Verzeichnis, in dem Sie Ihre Rexx-Programme speichern möchten, z.B. C:\USRCMD.
- Bearbeiten Sie AUTOEXEC.LSL: fügen Sie die Zeile "SETENV Rexx_MACROS C:\USRCMD; C:\LSLCMD" ein.
- Kopieren Sie Ihr Rexx-Programm in C:\USRCMD.
- Führen Sie das Rexx-Programm aus (CLI-Befehl: Rexx <Name des Rexx-Programms>).

Ausführung von Rexx-Programmen

Rexx-Programme werden durch das Eintragen des folgenden CLI-Befehls ausgeführt:

Rexx [<program>]

Wobei <program> der Name des Rexx-Programms ist, das ausgeführt werden soll. Wenn kein Programm angegeben ist, wartet LasalOS-Rexx auf den Eingabe von Rexx-Befehlen und führt die Befehle aus, wenn das Ende-der-Datei-Zeichen (Strg-Z) eingegeben wird.

Externe Rexx-Programme

Das LasalOS-Rexx sucht mittels einer Kombination aus der Rexx_MACROS-Umgebungsvariable und den Dateinamenerweiterungen nach Rexx-Programmen. Diese Regel gilt sowohl für externe Funktionsaufrufe als auch dem in der Befehlszeile angegebenem Programm. Wenn der Benutzer eine Funktion aufgerufen hat, wird wie folgt codiert.

Call myextfunc arg1, arg2

Zuerst sucht LasalOS-Rexx nach einer Datei namens myextfunc im aktuellen Verzeichnis. Wird der Datei nicht gefunden, sucht er nach einer Datei mit dem Namen myextfunc in jedem Verzeichnis, das in der Rexx_MACROS Umgebungsvariable angegeben ist. Wenn die Datei nicht gefunden wird, versucht LasalOS-Rexx eine Datei mit dem Namen myextfunc.rex im aktuellen Verzeichnis zu finden. Danach wird jedes Verzeichnis in Rexx_MACROS durchsucht. LasalOS-Rexx führt dann mit dem Anhängen von .cmd an den angegebenen externen Funktionsnamen fort, die von .rx gefolgt wird. Nur wenn keine Datei mit dem angegebenen Dateinamen oder einem Dateinamen mit der Endung .rex, .rx oder .cmd vorhanden ist, zeigt LasalOS-Rexx eine Meldung, dass die externe Funktion nicht bekannt ist. Um die Umgebungsvariable zu setzen, verwenden Sie diesen CLI-Befehl SETENV.

SETENV <name of variable> [<value of variable>]

Beispiel

```
/* Sample Rexx program that displays a menu */

interface = 'IP'
address = '192.168.44.107'

Do While option <> 0

Say 'Online parameter of remote station: 'interface'/'address
Say ' 0 .. Exit'
Say ' 1 .. Change online parameter'
Say ' 2 .. GETCPUSTATUS - Display CPU-Status of a remote station'
Say 'Please enter a number:'

option = Linein()
Say 'option:' option

Select
  When option = 0 Then Nop
  When option = 1 Then Call read_onlpar
  When option = 2 Then Do
    retval = GETCPUSTATUS(interface,address)
    if retval >= 0 then
      Say 'CpuStatus of' interface'/'address 'is' retval
  else
    Say 'GETCPUSTATUS failed, retval=' retval
  End
  Otherwise Say 'invalid option'
End

If option <> 0 Then Do
  Say 'Press Enter to continue ...'
  enter = Linein()
End

End

Exit

/*-- read_onlpar: prompts the user to enter the online parameters ---*/
read_onlpar: Procedure Expose interface address

  Say 'Enter Interface-Name (CAN1/CAN2/IP):'
  interface = Linein()
  Say 'Enter Address (CAN:StationNbr, IP:w.x.y.z:[port]):'
  address = Linein()

  return
```

In [Anhang A](#) finden Sie weitere Beispiele von Rexx-Programmen.

1.6.3 Integrierte LASALOS-spezifische Funktionen

LasalOS Rexx verfügt über zusätzliche Funktionen zur Kommunikation mit einer Remote-SPS (Online-Funktionen) und anderen nützlichen Funktionen, die nicht Teil der Rexx Sprache sind.

Berücksichtigungen bei der Verwendung von Online-Funktionen

Jede Online-Funktion hat mindestens 2 Parameter: interface und address. Eine Remote-SPS wird durch die Definierung des Schnittstellentyps (CAN, IP, ..) und der Adresse angesprochen. Die CAN-Schnittstellen-Adresse ist die Stationsnummer (0-31), die IP-Schnittstellen-Adresse ist die IP-Nummer im Internet-Standard ". " (dotted).

Für jede Online-Funktion muss eine Online-Verbindung zur Remote-SPS durch einen Aufruf der Funktion [ONLINE\(\)](#) erstellt werden. Wird die Online-Verbindung nicht mehr benötigt, sollte [OFFLINE\(\)](#) aufgerufen werden, um die Ressourcen wieder freizugeben. [ONLINE\(\)](#) hat einen Online-Beschreiber, der als Parameter für die [ONLINE\(\)](#)-Funktion verwendet wird.

Die Verbindung kann auch erstellt werden, indem Sie den Schnittstellentyp und die Adresse direkt an die Online-Funktion senden, anstatt den Online-Beschreiber zu verwenden. In diesem Fall wird die Online-Verbindung in der Online-Funktion aktiviert und deaktiviert. Der Vorteil dieser Methode ist, dass der Aufrufer die [ONLINE\(\)](#)- oder [OFFLINE\(\)](#)-Funktion aufrufen muss. Der Nachteil ist jedoch, dass für jede neue Online-Verbindung ein Online-Befehl erstellt werden muss und viel Ressourcen benötigt werden. Diese Methode sollte verwendet werden, wenn ein Online-Befehl in regelmäßigen Abständen aufgerufen wird (zum Beispiel, wenn der CPU-Status abgefragt wird).

Der interface und address Parameter kann die folgenden Werte enthalten:

Schnittstelle	Adresse
CAN1 oder CAN	<CAN Stationsnummer> (0-31)
CAN2 oder CAN	<CAN Stationsnummer> (0-31)
IP1 oder IP	<IP-nummer> (Dotted decimal-Format)
DESCR	<Descriptor-nummer Rückkehr von ONLINE>



Einige Befehle akzeptieren einen lokale Schnittstellentyp. In diesem Fall ist das Ziel die lokale SPS und nicht die Remote-SPS. Der Adressparameter für eine LOCAL-Schnittstelle ist ein beliebiger String (kein leerer String).

Beispiel

```

descr_num = ONLINE('CAN1',25)
status = GETCPUSTATUS('DESCR', descr_num)
call OFFLINE(descr_num)

status = GETCPUSTATUS('IP', '192.168.44.1')

retval = GETDATAD('LOCAL','N/A',748,2) /* 748 = 0x2EC = OS Version */

```

1.6.3.1 DIRLIST

DIRLIST wird verwendet, um den Inhalt eines Verzeichnisses aufzulisten.

Übergabeparameter		Typ	Beschreibung
Verzeichnis			Name des Verzeichnisses
stem			Ausgabe in der Stem der Compound-Variable
Rückgabeparameter		Typ	Beschreibung
			0 Funktion erfolgreich
			#0 Fehler-Code

Variable stem

<stem>.0	Anzahl der Verzeichniseinträge
<stem>.1-<stem>.n	Verzeichniseinträge.
<stem>.NAME	Name der abgestimmten Datei/Verzeichnis, ohne dem Pfad
<stem>.SIZE	Länge der Datei in Byte.
<stem>.TIME_WRITE	Zeitpunkt das letzte Schreiben an der Datei.
<stem>.TIME_CREATE	Zeit der Dateierstellung (-1L für FAT-Dateisysteme)
<stem>.TIME_ACCESS	Zeitpunkt des letzten Dateizugriffs (-1 für FAT-Dateisysteme)
<stem>.A_ARCH	Archiv; wird immer eingestellt, wenn die Datei geändert wurde und vom BACKUP-Befehl gelöscht.
<stem>.A_HIDDEN	Versteckte Datei; nicht normalerweise mit dem DIR-Befehl gesehen, außer wenn der /AH Option verwendet wird; liefert Informationen über die normalen Dateien sowie Dateien mit

	diesem Attribut zurück.
<code><stem>.A_NORMAL</code>	Normal; Datei kann gelesen oder beschrieben werden, ohne Einschränkungen.
<code><stem>.A_RDONLY</code>	Nur-Lese-Modus; Datei kann nicht zum Schreiben geöffnet werden und eine Datei mit dem gleichen Namen kann nicht erstellt werden.
<code><stem>.A_SUBDIR</code>	Unterverzeichnis
<code><stem>.A_SYSTEM</code>	Systemdatei; Nicht normalerweise mit dem DIR-Befehl gesehen, außer wenn der /A oder /A:S Option verwendet wird.

1.6.3.2 DRIVEINFO

DriveInfo wird verwendet, um Informationen über ein Laufwerk auf einer Remote-Station aufzulisten.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen).
Drive		Name des Laufwerks
stem		Gibt den Namen der Variable an, in welche die Informationen geschrieben werden
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich ≠0 Fehler-Code

Die Variable stem enthält folgende Parameter

Variable	Beschreibung
stem.BYTES_PER_SECTOR	Anzahl der Bytes pro Sektor
stem.SECTORS_PER_CLUSTER	Anzahl der Sektoren pro Cluster
stem.TOTAL_CLUSTERS	Anzahl der gesamten Cluster
stem.FREE_CLUSTERS	Anzahl der freien Cluster
stem.TOTAL_BYTES	Größe des Laufwerkes in Byte
stem.USED_BYTES	Größe des belegten Speichers in Byte
stem.FREE_BYTES	Größe des freien Speichers in Byte

Anforderungen

Das Rexx-Kommando wird ab der Version 01.01.024 der Rexx.dlm unterstützt. In Salamander erfolgt die Unterstützung des Kommandos ab der Betriebssystemversion 09.03.080 bzw. 09.02.020.

1.6.3.3 FILEINFO

FileInfo wird verwendet, um Informationen über eine Datei auf einem Remote-Station aufzulisten. Die Ausgabe wird in die Compound-Variablen stem geschrieben.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen).
filename		Name der Datei
stem		Ausgabe in der Stem der Compound-Variable
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich ≠0 Fehler-Code

Variable stem

<code><stem>.NAME</code>	Name der abgestimmten Datei/Verzeichnis, ohne dem Pfad
<code><stem>.SIZE</code>	Länge der Datei in Byte.
<code><stem>.TIME_WRITE</code>	Zeitpunkt das letzte Schreiben an der Datei.
<code><stem>.TIME_CREATE</code>	Zeit der Dateierstellung (-1L für FAT-Dateisysteme)
<code><stem>.TIME_ACCESS</code>	Zeitpunkt des letzten Dateizugriffs (-1 für FAT-Dateisysteme)
<code><stem>.A_ARCH</code>	Archiv; wird immer eingestellt, wenn die Datei geändert wurde und vom BACKUP-Befehl gelöscht.
<code><stem>.A_HIDDEN</code>	Versteckte Datei; nicht normalerweise mit dem DIR-Befehl gesehen, außer wenn der /AH Option verwendet wird; liefert Informationen über die normalen Dateien sowie Dateien mit diesem Attribut zurück.

<code><stem>.A_NORMAL</code>	Normal; Datei kann gelesen oder beschrieben werden, ohne Einschränkungen.
<code><stem>.A_RDONLY</code>	Nur-Lese-Modus; Datei kann nicht zum Schreiben geöffnet werden und eine Datei mit dem gleichen Namen kann nicht erstellt werden.
<code><stem>.A_SUBDIR</code>	Unterverzeichnis
<code><stem>.A_SYSTEM</code>	Systemdatei; Nicht normalerweise mit dem DIR-Befehl gesehen, außer wenn der /A oder /A:S Option verwendet wird.

1.6.3.4 GETCH

Erhält ein Zeichen von der Konsole ohne Echo.

Bemerkungen

Die Funktion GETCH() liest ein einzelnes Zeichen von der Konsole ohne Echo. Beim Lesen einer Funktionstaste oder eine Pfeiltaste, muss GETCH() zweimal aufrufen werden. Der erste Anruf liefert 0 zurück, und der zweite Aufruf liefert den eigentlichen Key-Code zurück.

Return Value

Diese Funktion gibt den ASCII-Code des Zeichens zurück.

1.6.3.5 GETCHE

Erhält ein Zeichen von der Konsole mit Echo.

Bemerkungen

Die Funktion GETCHE() liest ein einzelnes Zeichen von der Konsole aus und spiegelt den Charakter lesen. Beim Lesen einer Funktionstaste oder eine Pfeiltaste, muss GETCHE() zweimal aufgerufen werden. Der erste Anruf liefert 0 zurück, und der zweite Aufruf liefert den eigentlichen Key-Code zurück.

Return Value

Diese Funktion gibt den ASCII-Code des Zeichens zurück.

1.6.3.6 GETCPUSTATUS

Ruft die CPU-Status einer Remote-Steuerung.

Übergabeparameter		Typ	Beschreibung
Schnittstelle			
Adresse			Adresse der SPS (siehe Verwendung von Online-Funktionen).
Rückgabeparameter		Typ	Beschreibung
			≥ 0 CPU Status-Code <0 Fehler-Code

1.6.3.7 GETDATAD

Ruft Daten aus dem Applikationsdatenbereich einer lokalen oder Remote-Steuerung.

Übergabeparameter		Typ	Beschreibung
Schnittstelle			Schnittstellentyp LOCAL wird unterstützt
address			Adresse der SPS (siehe Verwendung von Online-Funktionen)
mem-addr			Memory-Adresse im Vergleich zu Beginn der Applikationsdatenbereich
length			Länge des ersuchten Speicher, nur Werte von 1 bis 256 sind erlaubt
stem			Ausgabe in der Stem der Compound-Variable. <stem>. len enthält die Länge, <stem>. 0 das erste Byte und so weiter.
Rückgabeparameter		Typ	Beschreibung
			0 Funktion erfolgreich $\neq 0$ Fehler-Code

1.6.3.8 GETOSVERSION

Erhält die Versionsnummer des Betriebssystems. Die Verwendung dieser Funktion ist veraltet, verwenden Sie stattdessen die Funktion GETOSVERSION2.

Übergabeparameter	Typ	Beschreibung
-------------------	-----	--------------

interface		Schnittstellentyp LOCAL wird unterstützt
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
Rückgabeparameter	Typ	Beschreibung
		<p>Tritt kein Fehler auf, liefert die Funktion den CPU-Status-Code zurück. Andernfalls wird ein negativer Fehler-Code zurückgegeben. Die Versionsnummer wird in einem Dezimal-Format zurückgegeben, z. B. OS Version 5.42 wird als der Wert 542 zurückgeliefert.</p> <p>Die Formel für den zurückgelieferten Wert ist wie folgt: $ver_major * 100 + ver_minor$</p>



Wegen des neuen 3-stelligen Formats der OS-Versionsnummer, liefert diese Funktion keine eindeutigen Werte, z. B.:

OS-Version 5.44 -> Rückgabewert = 544

OS-Version 1.1.44 -> Rückgabewert = 1744

OS-Version 1.1.101 -> Rückgabewert = 1801

OS-Version 1.2.001 -> Rückgabewert = 1801

1.6.3.9 GETOSVERSION2

Erhält die Versionsnummer des Betriebssystems.

Übergabeparameter	Typ	Beschreibung
interface		Schnittstellentyp LOCAL wird unterstützt
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
Rückgabeparameter	Typ	Beschreibung
		<p>Tritt kein Fehler auf, liefert die Funktion den CPU-Status-Code zurück. Andernfalls wird ein negativer Fehler-Code zurückgegeben. Die Versionsnummer wird in einem Dezimal-Format zurückgegeben, z. B. OS Version 5.42 wird als der Wert 542 zurückgeliefert.</p> <p>Die Formel für den zurückgelieferten Wert ist wie folgt: $(ver_major > 4) * 100000 + (ver_major & 0xf) * 1000 + ver_minor$</p>

Beispiel

OS-Version 5.44 -> Rückgabewert = 544

OS-Version 1.1.44 -> Rückgabewert = 101044

OS-Version 1.1.101 -> Rückgabewert = 101101

OS-Version 1.2.001 -> Rückgabewert = 102001

1.6.3.10 GETRES

Erhält die grafische Auflösung der CPU.

Rückgabeparameter	Typ	Beschreibung
		<p>Die zurückgegebene Text besteht aus zwei verschiedene Werte, die durch ein Komma "," getrennt sind.</p> <p>1. die Auflösung 2. die Rotation</p> <p>Wobei die Auflösung kann einer der folgenden werden</p> <p>VESA mode:</p> <p>GNO: Keine Anzeige G0: Textmode G10:Graphicmode 320x200x16 G11:Graphicmode 320x200x256 G12:Graphicmode 320x200x16M G20:Graphicmode 640x480x16 G21:Graphicmode 640x480x256 G22:Graphicmode 640x480x16M G30:Graphicmode 800x600x16 G31:Graphicmode 800x600x256 G32:Graphicmode 800x600x16M G40:Graphicmode 1024x768x16 G41:Graphicmode 1024x768x256 G42:Graphicmode 1024x768x16M G50:Graphicmode 1280x1024x16 G51:Graphicmode 1280x1024x256 G52:Graphicmode 1280x1024x16M</p> <p>Sigmatek mode:</p> <p>128x64: 128x64x1</p> <p>Und Rotation wird durch eine "norot" (ohne Rotation) oder "<direction Winkel>" (für beliebige Drehung) definiert.</p> <p>Richtung: CW oder CCW</p> <p>Winkel: Z.B.: 90</p> <p>So, CCW90 bedeutet ein Gegen-Uhrzeigersinn Drehung um 90 °.</p>

1.6.3.11 HWINFO

Stellt Informationen über die Steuerung bereit.

Übergabeparameter	Typ	Beschreibung
Info_dst		Gibt den Namen der Variable an, in welche die Informationen geschrieben werden
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich ≠0 Fehler-Code

Ansonsten wird ein Fehler-Code zurückgegeben. Die Variable `info_dst` enthält die folgenden Parameter:

Variable	Beschreibung
<code>Info_dst.PLATFORM</code>	Der Name der Plattform, z.B. „C-IPC AMD LX800“
<code>Info_dst.FPGAVER</code>	Die FPGA-Version
<code>Info_dst.SERIAL</code>	Seriennummer des Moduls
<code>Info_dst.OSVERSION</code>	Version des Betriebssystems.
<code>Info_dst.FPGA_PRODUCTID</code>	Produkt-ID des FPGA
<code>Info_dst.TARGET</code>	Gibt die Prozessorarchitektur der Plattform zurück

Die in der oberen Tabelle angegebenen Elemente der Struktur existieren ab der Rexx Version 01.01.020.

1.6.3.12 HWINFOONLINE

Stellt Informationen über die angegebene Remote-Steuerung bereit.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
Info_dest		Gibt den Namen der Variable an, in welche die Informationen geschrieben werden
Rückgabeparameter	Typ	Beschreibung

		0 Funktion erfolgreich
		#0 Fehler-Code

Die Variable `info_dst` enthält folgende Parameter:

Variable:	Beschreibung
<code>Info_dst.PLATFORM</code>	Name der Plattform, z.B „C-IPC AMD LX800“
<code>Info_dst.FPGAVER</code>	Die FPGA-Version
<code>Info_dst.SERIAL</code>	Seriennummer des Geräts
<code>Info_dst.OSVERSION</code>	Version des SPS-Betriebssystems.
<code>Info_dst.FPGA_PRODUCTID</code>	Produkt-ID des FPGA
<code>Info_dst.TARGET</code>	Gibt die Prozessorarchitektur der Plattform zurück

Anforderungen

Das Rexx-Kommando wird ab der Version 01.01.024 der Rexx.dlm unterstützt. In Salamander erfolgt die Unterstützung des Kommandos ab der Betriebssystemversion 09.03.080 bzw. 09.02.020.

1.6.3.13 KBHIT

Prüft, ob eine Taste im Tastaturpuffer vorhanden ist.

Rückgabeparameter	Typ	Beschreibung
		0 Keine Taste betätigt
		#0 Eine Taste wurde gedrückt

1.6.3.14 LSLLINK

Verbindet der übertragenen Module mit einem Remote-Steuerung.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)

Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		≠0 Fehler-Code

1.6.3.15 LSLLOADPRJ

Setzt alle übertragenen Module neu und stellt optional den Namen des Projekts in einem Remote-Steuerung ein.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
prjname		Projektname
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		≠0 Fehler-Code

1.6.3.16 LSLSEND

Zur Übertragung eines LASAL-Projekts an eine Remote-Steuerung.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
Idx-Datei		Gibt den Namen der Projektindex Datei an.
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		≠0 Fehler-Code

1.6.3.17 LSLSENDMOD

Überträgt einem einzigen LASAL-Modul zu einem Remote-Steuerung.

Übergabeparameter		Typ	Beschreibung	
Schnittstelle				
Adresse			Adresse der SPS (siehe Verwendung von Online-Funktionen)	
lob-file			Dateiname der LASAL-Objektdatei	
Rückgabeparameter		Typ	Beschreibung	
			0 Funktion erfolgreich ≠0 Fehler-Code	

1.6.3.18 LSLSENDMODCLSCNAME

Diese Funktion überträgt ein einzelnes LASAL-Modul an eine Remote-Steuerung. Im Gegensatz zu der LSLSENDMOD-Funktion verwendet diese Funktion den Namen der Klasse und nicht die Modul-ID um das Modul in das Zielsystem zu identifizieren (Hinweis: Die Modul-ID besteht aus dem Projektnamen und dem Klassennamen). Diese Funktion kann verwendet werden, um eine Klasse aus einem Projekt zu ersetzen, deren Name nicht mit dem Namen des Projekts im Zielsystem übereinstimmt.

Übergabeparameter		Type	Beschreibung	
Schnittstelle				
Adresse			Adresse der SPS (siehe Verwendung von Online-Funktionen)	
lob-file			Dateiname der LASAL-Objektdatei	
Rückgabeparameter		Type	Beschreibung	
			0 Funktion erfolgreich ≠0 Fehler-Code	

1.6.3.19 MILLISLEEP

Hält für die angegebene Zahl von Millisekunden an.

Übergabeparameter	Typ	Beschreibung
milliseconds		Gibt die Zeit zur unterbrechen der Ausführung in Millisekunden an
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich ≠0 Fehler-Code

1.6.3.20 OFFLINE

Beendet eine Online-Verbindung zu einer Remote-SPS.

Übergabeparameter	Typ	Beschreibung
online-descriptor		Online-descriptor, der von der Funktion ONLINE zurückgegeben wird.
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich ≠0 Fehler-Code

1.6.3.21 ONLINE

Erstellt eine Online-Verbindung zu einer Remote-SPS.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen). Mögliche Optionen im Address-Sring. Die Optionen müssen durch ein Semikolon getrennt sein und dürfen keine Leerzeichen enthalten. Diese Optionen können bei allen Remote Kommandos verwendet werden. "ROUTE=1" "ROUTE_PWD=pass" "ONLINE_PWD=pass" Beispiel:

		ONLINE('IP','10.10.150.1;ROUTE=1;ROUTE_PWD=pass1;ONLINE_PWD=pass2')
Rückgabeparameter	Typ	Beschreibung
		Tritt kein Fehler auf, liefert die Funktion den Online-Descriptor zurück. Andernfalls wird ein negativer Fehler-Code zurückgegeben.

Verwenden Sie [Offline](#), um die Verbindung zu beenden. Ein Programm sollte immer einen passenden Aufruf an OFFLINE für jeden erfolgreichen Aufruf an ONLINE, um jeweiligen Verbindungsressourcen an dem System zurückzugeben.

Wenn eine TCP-/IP-Verbindung verwendet wird, beendet die abgesetzte Steuerung automatisch die Verbindung, wenn keine Daten innerhalb von 30 s (neuere Betriebssysteme verwenden einen Timeout Wert von 60 s) übertragen werden. Um eine TCP/IP-Verbindung zu führen, muss alle Funktionen, die eine Verbindung anfordert (z.B. [GetCpuStatus](#)) in regelmäßigen Zeitabständen aufgerufen.

1.6.3.22 RESETSPS

Setzt ein Remote-Steuerung neu.

Übergabeparameter		Typ	Beschreibung
Schnittstelle			
Adresse			Adresse der SPS (siehe Verwendung von Online-Funktionen)
Rückgabeparameter		Typ	Beschreibung
			0 Funktion erfolgreich
			#0 Fehler-Code

1.6.3.23 REXEC

Führt eine CLI-Kommando auf einem Remote-Steuerung aus.

Übergabeparameter		Typ	Beschreibung
Schnittstelle			
Adresse			Adresse der SPS (siehe Verwendung von Online-Funktionen)
Rückgabeparameter		Typ	Beschreibung

		0 Funktion erfolgreich
		#0 Fehler-Code



Wenn erfolgreich ausgeführt, wird der Befehl übertragen und initialisiert in der Remote-Steuerung. Der Rückgabecode für diesen Befehl ist nicht der Rückgabecode für das CLI-Kommando! Dieser Befehl wartet auch nicht, bis der CLI-Befehl abgeschlossen ist, es wird asynchron ausgeführt.

1.6.3.24 RUNSPS

Startet die Applikation auf einem Remote-Steuerung.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
Übergabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		#0 Fehler-Code

1.6.3.25 RXFILE

Überträgt eine Datei von einer über Ethernet verbundenen Remote-SPS zur aufrufenden Steuerung.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
fname_src		Gibt den Namen der Quelldatei an (Pfad+Name auf der Remote-Steuerung)
fname_dst		Gibt den Namen der Ziel-Datei an (Pfad+Name, den die Datei auf der empfangenden Steuerung haben soll)
Übergabeparameter	Typ	Beschreibung

			0 Funktion erfolgreich
			#0 Fehler-Code

1.6.3.26 SENDOS

Überträgt und installiert ein Betriebssystem in einem Remote-Steuerung. Nach dem das Betriebssystem installiert wurde, wird die SPS neugestartet.

Übergabeparameter		Typ	Beschreibung
Schnittstelle			
Adresse			Adresse der SPS (siehe Verwendung von Online-Funktionen).
osimage-file			Name der Datei, die das Betriebssystem Bild enthält. diese Datei hat normalerweise die Erweiterung. BIN oder. RTB
Rückgabeparameter		Typ	Beschreibung
			0 Funktion erfolgreich
			#0 Fehler-Code
			Ein Fehler das Re-Booten der SPS ist kein Fehler.

1.6.3.27 SERVERREAD

ServerRead liefert einen Serverwert. Es können sowohl lokale Server als auch Server von Remote CPUs gelesen werden. Server können nur gelesen werden, wenn sich die CPU im "RUN" befindet.

Übergabeparameter		Typ	Beschreibung
Schnittstelle			
Adresse			Adresse der SPS (siehe Verwendung von Online-Funktionen)
servername			Name des Servers
stem			Name der Variable, auf die der Serverwert geschrieben wird
Rückgabeparameter		Typ	Beschreibung
			0 Funktion erfolgreich

		#0 Fehler-Code
--	--	--------------------------------

1.6.3.28 SERVERWRITE

Mit ServerWrite können Daten auf einen Server geschrieben werden. Es können sowohl lokale Server als auch Server von Remote CPU's beschrieben werden. Server können nur beschrieben werden, wenn sich die CPU im "RUN" befindet.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
servername		Name des Servers
stem		Neuer Serverwert
Übergabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		#0 Fehler-Code

1.6.3.29 SETBREAKPOINT

Setzt einen Breakpoint in der SPS.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
nr		Breakpoint-Nummer (0-3)
addr		Addr Code-Adresse des Breakpoints
len		Länge der Breakpoint (nur für Daten-Breakpoints)
type		Typ der Breakpoint <ul style="list-style-type: none"> 0 Instruction breakpoint 1 Instruction breakpoint AWL 2 Data Breakpoint schreiben

		3 Daten-Breakpoint schreiben+lesen
stackOfs		Offset des vom Betriebssystem gespeicherten Stack-Bereichs, wenn die CPU bei einem Breakpoint hält
stacksize		Größe des vom Betriebssystem gespeicherten Stack-Bereichs, wenn die CPU bei einem Breakpoint hält
cntr		Wie oft der Breakpoint vorkommen muss, bevor die CPU angehalten wird
code		Code für einen bedingten Breakpoint
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich ≠0 Fehler-Code

1.6.3.30 TXFILE

Überträgt eine Datei in einer Remote-SPS

Übergabeparameter		Typ	Beschreibung
Schnittstelle			
Adresse			Adresse der SPS (siehe Verwendung von Online-Funktionen).
fname_src			Gibt den Namen der Quelldatei an.
fname_dst			Gibt den Namen der Ziel-Datei an, z. B. der Name der Datei auf dem Remote-Steuerung. Wenn die angegebene Datei bereits auf dem Remote-Steuerung existiert, werden ihre Inhalte zerstört.
Rückgabeparameter	Typ	Beschreibung	
		0 Funktion erfolgreich ≠0 Fehler-Code	

1.6.4 Command Line Interface (CLI) Befehle

Sie können Befehle aus einem Rexx-Programm an die Befehlszeilenschnittstelle (CLI) senden, indem Sie entweder einen Ausdruck angeben, der als Befehl ausgeführt werden soll, oder indem Sie die Anweisung ADDRESS mit einem Umgebungsnamen von SYSTEM verwenden.

Nachdem der CLI den Befehl ausgeführt hat, wird die besondere Variable RC auf den Rückgabewert des CLI-Befehls eingestellt. In der Regel geben alle CLI-Kommandos einen Wert von 0 zurück, wenn keine Fehler auftreten.

Beispiel

```
/* Sending a command to the Command Line Interface by
 * issuing an expression executed as a command.
 */
Say 'Enter CLI command:'
cmd = Linein()
cmd
Say 'return code of CLI command: 'rc
/* Sending a command to the Command Line Interface by
 * using the instruction ADDRESS.
 */
Say 'Enter CLI command:'
cmd = Linein()
Address System cmd
Say 'return code of CLI command: 'rc
```

1.6.5 Anhang A - Beispiel Rexx-Programme

VERSION

Das VERSION Programm sammelt verschiedene Systeminformationen und gibt sie auf dem Bildschirm aus. Es ist hilfreich, um herauszufinden, welche Version von Rexx auf dem System installiert ist.

```
/* VERSION:
 * Collect various system information and print it on the screen.
 */
Parse Source platform invocation filename

Say
Say 'Source:'
Say '  platform      : 'platform
Say '  invocation    : 'invocation
Say '  filename      : 'filename

Parse Version language level date month year
```

```

Say
Say 'Rexx Version:'
Say '  language      : 'language
Say '  version       : 'level
Say '  time          : 'date month year

Rexx_macros = getenv('Rexx_MACROS')

Say
Say 'Envrionment:'
Say '  Rexx_MACROS  : 'Rexx_macros

```

Automatischer Programmstart von austauschbaren Medien

Das folgende Verfahren zeigt, wie man ein System zum Suchen vorbereitet und ein Rexx AUTOSTRT Programm auf einem austauschbaren Datenträger (z.B. USB-Stick) ausführt. Dieses Verfahren kann zur Aktualisierung der Software auf einer SPS mit minimalem Benutzereingriff verwendet werden. Die Medien müssen nur eingefügt werden, wo AUTOSTRT liegt und die SPS neu gestartet. In diesem Beispiel kopiert das AUTOSTRT Programm die Dateien aus den austauschbaren Medien in ein Verzeichnis auf der SPS.

Erste Aktualisierung der AUTOEXEC.LSL mit den folgenden Angaben:

```

SETENV Rexx_MACROS B:\;F:\;G:\;H:\;I:\;J:\;K:\;L:\;M:\;N:\;O:\;
Rexx AUTOSTRT
SETENV Rexx_MACROS C:\LSLCMD

```

Die erste Zeile setzt das Programm Suchpfad zu einer Liste der Laufwerke mit austauschbaren Medien, die nächste Zeile versucht dann, das Rexx AUTOSTRT Programm aus einem Verzeichnis zu starten, welches im vorherigen Schritt angegeben wurde. Dieses Programm wird ausgeführt, wenn beim Hochfahren ein Wechseldatenträger mit AUTOSTRT in die SPS eingelegt wird; andernfalls wird eine Fehlermeldung (Programm nicht gefunden) angezeigt. Die letzte Zeile setzt den Programmsuchpfad zurück.

Der folgende Text ist ein Beispiel für ein AUTOSTRT Programm:

```

/*
-----*
 AUTOSTRT - copy files from one directory to another and reboot PLC
-----*/
Parse Source . . filename

/* query the drive letter of the removable media */
drive = LEFT(filename,2)

/* assign source- and destination directory names to variables */
source_dir = drive'\SRC'
dest_dir   = 'C:\TESTDIR'

If (stream(dest_dir,'C','QUERY EXISTS') = '') Then Do

```

```
/* destination directory does not exist -> create it */
cmd = 'MKDIR 'dest_dir
Address System cmd
If rc \= 0 Then Do
    Say cmd' failed, rc='rc
    Call fatal_error
End
End

/* copy all files from the source directory on the removable media
* to the destination directory
*/
cmd = 'XCOPY 'source_dir'\*.* 'dest_dir
Address System cmd
If rc \= 0 Then Do
    Say cmd' failed, rc='rc
    Call fatal_error
End
End

Call reboot_plc

/* end of program */
Exit

/*-----
 Procedures
-----*/
/* ----- */
reboot_plc: Procedure

    Parse Source . . filename
    drive = LEFT(filename,2)
    Say filename' executed successfully'
    Say '*****'
    Say '*** Remove media in drive 'drive' and reboot ***'
    Say '***'
    Say '*****'
    Do Forever
        Nop
    End

    Return

/* ----- */
fatal_error: Procedure

    Parse Source . . filename
    Say 'Fatal error while executing 'filename
    Say '*****'
    Say '***'
    Say '*** System halted ***'
    Say '***'
    Say '*****'
```

```
Do Forever
  Nop
End

Return
```

1.6.6 Anhang B - Fehler-Codes

Die nachfolgende Tabelle listet die Fehler-Codes auf, welche von REXX-Befehlen ausgegeben werden können.

Fehler-Code	Beschreibung
0	Kein Fehler
-200	Initialisierung fehlgeschlagen
-201	Ungültiger Online-Typ
-202	Adresse ist bereits in Verwendung
-203	Zu wenig Speicher
-204	Keine Schnittstelle verfügbar
-205	Zu wenig Byte empfangen
-206	Zeitüberschreitung beim Warten auf einen CPU-Status
-207	Ungültige Adresse
-208	Systemfehler
-209	Maximale Verbindungsanzahl erreicht
-210	Falsche DESCR-Nummer
-211	Ungültige Parameter
-300	Socket konnte nicht erstellt werden
-301	Verbindungsaufbau fehlgeschlagen
-302	Ungültige IP-Adresse
-303	TCP/IP Fehler
-304	Timeout
-305	Pufferspeicher ist zu klein
-306	Verbindung wurde geschlossen
-307	Negatives Kommando empfangen

-308	Ungültiges Format empfangen
-309	Kein Header gefunden
-310	Logon fehlgeschlagen

1.6.7 Anhang C - REXX How To

In diesem Anhang werden grundlegende Funktionen von REXX anhand von Beispielen erklärt, um Programmierern in Zukunft die Arbeit zu erleichtern.

1.6.7.1 Funktionsaufruf

Interne Funktionen werden über ein Label gekennzeichnet. Wenn nur ein Label die Funktion kennzeichnet, hat diese Funktion auf alle Variablen Zugriff. Wenn hinter dem Label das Keyword „Procedure“ steht, sind alle Variablen unzugänglich für diese Funktion. Mittels eines „Procedure Expose“ und anschließend dem Variablenamen, können ausgewählte Variablen für diese Funktion zugänglich gemacht werden.

Eine Funktion, die keinen Wert zurückgibt, muss über den „call“-Parameter aufgerufen werden. Bei der Argumentübergabe können in diesem Falle die Klammern weggelassen werden. Siehe Code Beispiel Ausgabe 1 und 3. Falls die aufgerufene Funktion einen Rückgabeparameter hat, kann dieser über die Variable: „RESULT“ abgerufen werden.



Wenn eine Funktion eine Funktion aufruft und die Unterfunktion braucht Zugriff auf eine Variable, dann muss diese Variable bei beiden Funktionen freigegeben werden.



Wenn eine Funktion aufgerufen wird, ist es insbesondere bei x86-CPUs notwendig, den Rückgabewert abzufragen. Ansonsten kann es zu unerwartetem Verhalten kommen.

```
a = 10
b = 50
erg = 0

/*Methode Addition wird aufgerufen*/
call addition
```

```
say 'Ausgabe1: 'erg
_____
/*Methode Subtraction wird aufgerufen*/
erg = subtraction(b,a)
say 'Ausgabe2: 'erg
_____
/*Methode Multiplication wird aufgerufen*/
call multiplication a, b
say 'Ausgabe3: 'erg
_____
/*Methode Division wird aufgerufen*/
call division a, b
say 'Ausgabe4: 'RESULT
_____
return
_____
/*Diese Funktion kann auf alle Variablen zugreifen.*/
addition :
  erg = a + b
  return
_____
/*Diese Funktion hat keinen Zugriff auf externe
Variablen.*/
subtraction : Procedure
  Parse Arg a, b
  return a - b
_____
/*Dies Funktion hat Zugriff auf die Variable erg*/
multiplication : Procedure Expose erg
  Parse Arg a, b
  erg = a * b
  return
```

```
division : Procedure
Parse Arg a, b
return a / b
```

1.6.7.2 Bedingungen und Schleifen

Bei Bedingungen und Schleifen ist es möglich eine kurze und eine lange Variante zu schreiben. Wenn nur eine Zeile ausgeführt werden soll, kann die kurze Variante verwendet werden, sollen aber mehrere Zeilen ausgeführt werden, muss die lange Variante verwendet werden.

```
rc = 1

if (rc == 1) then
  Say 'Alles ist gut gelaufen.'
Else
  Say 'Ein Fehler ist aufgetreten.'

if (rc <> 0) then do
  Say 'Ein Fehler ist aufgetreten.'
  Say 'Bitte neustarten!'
End
Else
  Say 'Alles ist gut gelaufen.'

if (rc == 25) then
  Say 'Der Rückgabewert ist 25'
Else do
  Say 'Der Rückgabewert ist ungleich 25'
  Say 'Es ist ein Fehler aufgetreten!'
End
```

Bei der kurzen Variante wird das „do“ nach dem „then“ weggelassen und dadurch wird kein „End“ benötigt. Bei der langen Variante wird beides benötigt. Oben im Beispiel sind verschiedene Varianten und Kombinationen gezeigt. Dasselbe gilt auch für Schleifen.

1.6.7.3 Online-Funktionen

Dies sind Funktionen, die eine Verbindung zu einer Remote-CPU aufbauen. Grundsätzlich werden dafür zwei Parameter: interface und address, benötigt. Für jede Online-Funktion muss eine Verbindung zur Remote-CPU durch einen Aufruf der Funktion [ONLINE](#) erstellt werden. Wird die Online-Verbindung nicht mehr benötigt, sollte die Funktion [OFFLINE](#) aufgerufen werden, um die Ressourcen wieder freizugeben. „ONLINE“ hat einen Online-Beschreiber, der als Parameter für die Online-Funktion verwendet wird.

Die Verbindung kann aber auch erstellt werden, indem der Online-Funktion direkt der Schnittstellentyp und die Adresse übergeben wird. Der Vorteil ist, dass man hier nicht mehr die Funktionen „ONLINE“ und [OFFLINE\(\)](#) aufrufen muss. Der Nachteil ist, dass für jede aufgerufene Online-Funktion, eine neue Online-Verbindung aufgebaut wird und erst nach einer gewissen Zeit wieder freigegeben wird. Dies braucht viele Ressourcen, die auch irgendwann aufgebraucht sind.

Es ist zu empfehlen, dass Online-Funktionen immer über den Aufruf der Funktion [ONLINE](#) verwendet werden.

Der interface und address Parameter kann die folgenden Werte enthalten:

Interface	Address
CAN1 oder CAN	<CAN Stationsnummer> (0-31)
CAN2 or CAN	<CAN Stationsnummer> (0-31)
IP1 oder IP	<IP-nummer> (Dotted decimal-Format)
DESCR	<Descriptor-nummer Rückgabe von ONLINE>



Einige Befehle akzeptieren einen lokalen Schnittstellentyp (LOCAL). In diesem Fall ist das Ziel die lokale CPU und nicht die Remote-CPU. Der Adressparameter für eine LOCAL-Schnittstelle ist ein beliebiger String (kein leerer String).

1.6.7.3.1 ONLINE(interface, address)

Erstellt eine Online-Verbindung zu einer Remote-CPU.

Übergabeparameter	Typ	Beschreibung
Schnittstelle		
Adresse		Adresse der SPS
Rückgabeparameter	Typ	Beschreibung
		Tritt kein Fehler auf, liefert die Funktion den Online-Descriptor zurück. Ansonsten gibt sie einen negativen Fehler-Code zurück.

1.6.7.3.2 OFFLINE(online-descriptor)

Beendet eine Online-Verbindung zu einer Remote-CPU.

Übergabeparameter	Typ	Beschreibung				
online-descriptor		Online-Descriptor, der von der Funktion ONLINE() zurückgegeben wird.				
Rückgabeparameter	Typ	Beschreibung				
		<table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td style="padding: 2px 5px;">0</td> <td style="padding: 2px 5px;">OK</td> </tr> <tr> <td style="padding: 2px 5px;">≠0</td> <td style="padding: 2px 5px;">Fehler-Code</td> </tr> </table>	0	OK	≠0	Fehler-Code
0	OK					
≠0	Fehler-Code					

1.6.7.3.3 Beispielfunktion

```

Desthandle = 
online_plc( 'xxx.xxx.xxx.xxx' ,YY)
If (desthandle > 0) then do
  rc = REXEC( 'DESCR' , desthandle , 'SETENV
  FILENAMES LONG' )
  if (rc == 0) then do
    say 'CMD succeeded.'
  End
Else do

```

```
    Say 'CMD failed.'  
End  
    retval = OFFLINE(desthandle)  
End  
  
online_plc: Procedure  
/*Retrieving function parameters*/  
Parse arg cpuip, maxretries  
  
retrycounter = 1  
desthandle = ONLINE( 'IP', cpuip )  
/*online connection failed*/  
if (desthandle < 0) then do  
    /*try going online again*/  
    do while ((retrycounter <= maxretries) & (desthandle < 0))  
        say 'Online connection failed:  
'retrycounter' try, retcode: 'desthandle  
        desthandle = ONLINE( 'IP',  
cpuip )  
  
    retrycounter = retrycounter + 1  
    retval = MILLISLEEP(1000)  
end  
end  
  
/*online connection failed*/  
if (desthandle < 0) then do
```

```
        say 'Online connection failed,  
'cpuiip' not available('desthandle').'  
end  
else do  
    say 'Online connection to CPU  
'cpuiip'" succeeded.'  
end  
return desthandle
```

1.6.7.4 CLI-Kommandos ausführen

Bei CLI-Kommandos gibt es Unterschiede, ob sie auf der lokalen CPU ausgeführt werden oder auf einer Remote-CPU. Weiteres gibt es Unterschiede, wenn sie von einer ARM- oder x86-CPU ausgeführt werden.

1.6.7.5 Address System cmd

Dies führt ein CLI-Kommando auf der lokalen CPU aus. Der gezeigte Befehl setzt die Variable „FILENAMES“ auf „LONG“.

Wenn ein CLI-Kommando auf der lokalen CPU ausgeführt werden soll, darf KEIN „EXEC“ vor dem Befehl stehen.

```
cmd = 'SETENV FILENAMES LONG'
```

Address System cmd

1.6.7.5.1 REXEC(INTERFACE, ADDRESS, CMD)

Dieser Befehl führt ein Kommando auf einer Remote-CPU aus. Falls die lokale CPU den Typ „ARM“ hat, muss ein „EXEC“ dem Befehl vorgestellt sein (cmdARM) falls es eine „x86“-CPU ist, darf kein „EXEC“ davor sein (cmdX86).

```
cmdX86 = 'SETENV FILENAMES LONG'  
cmdARM = 'EXEC SETENV FILENAMES LONG'  
rc = REXEC( 'INTERFACE', 'ADDRESS', cmdX86 )  
rc = REXEC( 'INTERFACE', 'ADDRESS', cmdARM )
```

Übergabeparameter		Typ	Beschreibung
Schnittstelle			Schnittstellentyp LOCAL wird unterstützt
Adresse			Adresse der SPS (siehe Verwendung von Online-Funktionen)
cmd			Das Kommando, welches auf der Remote-CPU ausgeführt werden soll
Rückgabeparameter		Typ	Beschreibung
			0 Kein Fehler ≠0 Fehler-Code

Wenn erfolgreich ausgeführt, wird der Befehl übertragen und initialisiert in der Remote-Steuerung. Der Rückgabecode für diesen Befehl ist nicht der Rückgabecode für das CLI-Kommando! Dieser Befehl wartet auch nicht, bis der CLI-Befehl abgeschlossen ist, es wird asynchron ausgeführt.

1.6.7.5.2 CLI-Kommando: SHOWENV

Beim Aufruf des [SHOWENV](#)-Kommandos muss bei einer x86-CPU der Variablenname (z.B. FILENAMES) übergeben werden. Auf einer Salamander-CPU wird, egal ob mit oder ohne Variablenname, alles ausgegeben.

1.6.7.5.3 Pipe Operator „>“

Das Pipe-Kommando kann zum Weiterleiten von Ausgaben z.B. in eine Datei verwendet werden.

```

Parse Source . . filename
drive = LEFT(filename,2)
outfile = drive'\mac.txt'
cmd = 'ip >' outfile
Address System cmd
If rc \= 0 Then Do
  say 'Rediricting command IP to file FAILED'
End

```

Im Beispiel werden Netzwerkeinstellungen in einer Datei gespeichert. Aus dieser kann dann die IP, Mac Adresse etc. geparsst werden.

Diese Funktionalität wird unter Salamander nicht unterstützt.

1.6.7.5.4 GETDATAD(interface, address, mem-addr, length, stem)

Ruft Daten aus dem Applikationsdatenbereich einer lokalen oder Remote-Steuerung.

```
/*Auslesen der Systemvariablen _WhoAmI an der Stelle
M02E8
Mögliche Werte siehe lsl_st_kernel.h
(z.B. #define DESTPLC_C_IPC 0x00000020)*/

GetWhoAmI: Procedure
  Parse Arg interface, address

  retval = GETDATAD(interface,address,X2D('2E8'),4,mem)

  If retval = 0 Then Do
    whoami = mem.0 + mem.1 * 256 + mem.2 * 256 * 256 +
    mem.3 * 256 * 256 * 256
  End
  Else Do
    whoami = retval
  End

  Return whoami
```

Transfer parameters	Typ	Beschreibung
Schnittstelle		Schnittstellentyp LOCAL wird unterstützt.
Adresse		Adresse der SPS (siehe Verwendung von Online-Funktionen)
mem-addr		Memory-Adresse im Vergleich zu Beginn des Applikationsdatenbereichs
Length		Länge des ersuchten Speicher, nur Werte von 1 bis 256 sind erlaubt

Rückgabeparameter	Typ	Beschreibung
Stem		Die Ausgabe wird in den Stamm der zusammengesetzten Variablen geschrieben. <stem>.len zeigt die Länge an, <stem>.0 das erste Byte und so weiter.

1.6.7.6 Dateisystemoperationen

In diesem Teil wird das Lesen aus einer Datei und Schreiben in eine Datei beschrieben.

1.6.7.6.1 stream(stream, option, command)

Mit diesem Befehl hat man viele Möglichkeiten Dateioperationen durchzuführen. Einige wenige werden im Folgenden erklärt.

Öffnen einer Datei zum Schreiben

Dieser Befehl öffnet eine Datei. Dabei wird sie nicht überschrieben, sondern es wird nach dem bestehenden Inhalt geschrieben bzw. in eine neue Zeile.

Call	STREAM	'C:
\exampleTextFile.txt	'C'	'OPEN
APPEND		WRITE

Falls die Datei noch nicht besteht, wird sie erstellt. Soll der Inhalt einer bestehenden Datei überschrieben werden, muss anstelle von „OPEN WRITE APPEND“ „OPEN WRITE REPLACE“ übergeben werden.

Schließen einer Datei

Dieser Befehl schließt eine geöffnete Datei.

rc	=	stream('C:
\exampleTextFile.txt	'C'	'CLOSE')

Überprüfen, ob eine Datei/ein Ordner existiert

Dieser Befehl überprüft, ob der angegebene Pfad existiert. Wenn nicht gibt er einen leeren String zurück. Im Beispiel ist eine Überprüfung eingebaut.

```

rc = stream('C:\exampleTextFile.txt', 'C',
'QUERY EXISTS')
if (rc <> '')
say 'The file exists.'
Else
say 'The file does not exist.'

```

Größe einer Datei ermitteln

Dieser Befehl gibt die Größe der Datei in Byte zurück. Falls die Datei nicht existiert, wird ein leerer String zurückgegeben.

```

rc = stream('C:\exampleTextFile.txt', 'C', 'QUERY SIZE')
If (rc <> '') Then Do
  Say 'The File has a size of 'rc' Byte.'
End
Else
  Say 'Getting size of the file failed.'

```

1.6.7.6.2 lineout(stream, string, line)

Diese Funktion gibt den String in dem angegebenen Stream aus. In diesem Fall dient diese Funktion um in eine Datei zu schreiben. Die Datei muss hierfür bereits über den „stream“-Befehl geöffnet sein. Ist dies nicht der Fall, dann gibt die Funktion einen Fehler zurück.

```

string = 'Dieser Satz soll in die Datei
geschrieben werden.'

retcode = LINEOUT('C:\exampleTextFile.txt', string)
if (retcode <> 0) then do
  Call STREAM 'C:\exampleTextFile.txt', 'C', 'OPEN WRITE
APPEND'
  retcode = LINEOUT('C:\exampleTextFile.txt', string)
End

```

Im Beispiel überprüfen wir deshalb, ob es funktioniert hat, und wenn nicht, wird die Datei geöffnet.

Übergabeparameter	Typ	Beschreibung
stream		Stream, in den der String ausgegeben werden soll
string		String, der in den Stream geschrieben werden soll
line		Nummer der Zeile, in welche der String im Stream geschrieben werden soll
Rückgabeparameter	Typ	Beschreibung
		0 Schreiben war erfolgreich 1 Schreiben war nicht erfolgreich

1.6.7.6.3 linein(stream, line, count)

Diese Funktion liest im gezeigten Fall immer die nächste Zeile aus und speichert diese in der Variable „rc“. Wenn das Ende der Datei erreicht ist, beginnt sie von vorne. Wenn der Funktion keine Parameter übergeben werden, liest sie die Eingaben vom CLI-Interface ein.

```
rc = linein('C:\config.txt')
```

Übergabeparameter	Typ	Beschreibung
Stream		Stream, der eingelesen werden soll
Line		Zeilennummer, die eingelesen werden soll
Count		Anzahl der Zeilen, die eingelesen werden sollen
Rückgabeparameter	Typ	Beschreibung
		Funktion gibt Null zurück, wenn ein Fehler auftritt

1.6.7.6.4 lines(stream, option)

Diese Funktion gibt die verbleibende Anzahl der Zeilen in dem angegebenen Stream zurück

```
if (lines('C:\config.txt') == 0)
  Say 'Ende der Datei erreicht'
```

```

if (lines('C:\config.txt') == 25)
  Say 'Es befinden sich noch 25 Zeilen im Stream.'

if ((lines('C:\config.txt') == 1) | (lines('C:
\config.txt', 'N') == 1))
  Say 'Es befinden noch Eine oder mehr
Zeilen im Stream.'

```

Übergabeparameter	Typ	Beschreibung
Stream		Gibt den Stream an, von welchem die übrigbleibende Anzahl Zeilen ausgegeben werden soll.
Option		Hier kann gewählt werden, ob man die genaue Anzahl der Zeilen wissen will („C“) oder nur wissen will, ob noch Zeilen vorhanden sind („N“). Wenn kein Parameter übergeben wird, gibt sie entweder die genaue Anzahl der Zeilen zurück oder ob noch Zeilen vorhanden sind.
Rückgabeparameter	Typ	Beschreibung
		Siehe Beispiel

1.6.7.6.5 attrib(path, option)

Das Ändern von Datei und Ordner Attributen wird über CLI-Kommandos gemacht.

```

cmd = 'attrib C:\exampleFolder -r'
rc = REXEC('Interface', 'Address', cmd)
Say rc
Address System cmd
say rc

```

Dieser Befehl muss genau so wie in dem oben gezeigten Beispiel geschrieben werden, ansonsten funktioniert er entweder nur für x86-CPUs oder für ARM-CPUs.

Übergabeparameter	Typ	Beschreibung

Pfad		Pfad zu dem Ordner oder der Datei, dessen Attribute verändert werden sollen
option		<p>Hier wird das Attribut angegeben, welches verändert werden soll</p> <ul style="list-style-type: none"> + setzt ein Attribut - entfernt ein Attribut a Archiv Attribut h verstecktes Attribut r Read-only Attribut s System Attribut

1.6.7.7 REXX Fehler-Codes

In diesem Abschnitt werden Fehler-Codes ausführlicher erklärt.

1.6.7.7.1 -307 – Negatives Kommando empfangen

Dieser Fehler bedeutet, dass auf der Remote-CPU ein Fehler aufgetreten ist. Zum Beispiel kann dieser Fehler bei einem Kopiervorgang vorkommen, wenn kein Speicherplatz mehr auf der Remote-CPU vorhanden ist.

1.6.7.8 Nützliche Code-Abschnitte

1.6.7.8.1 Konvertieren

String in Groß-/Kleinbuchstaben konvertieren

```

upper = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'
lower = 'abcdefghijklmnopqrstuvwxyz'
stringOrg = 'This is a string with upper and lower case.'
say 'stringOrg: 'stringOrg'
stringTrans = translate(stringOrg, lower, upper)
say 'stringTrans (lower): 'stringTrans
stringTrans = translate(stringOrg)
say 'stringTrans (upper): 'stringTrans

```

1.6.7.8.2 Lesen und Verarbeiten

Datei Zeile für Zeile einlesen und verarbeiten

```
Parse Source . . filename
drive = Left(filename,2)
filePath = drive'\config.txt'
/*Überprüfen ob die Datei überhaupt existiert*/
rc = stream(filePath, 'C', 'QUERY EXISTS')
arrayLines.0 = ''
arrayWords.0 = ''
LinesCounter = 0
WordsCounter = 0
if (rc <> '') then do
    /*Überprüfen ob die Datei noch Zeilen hat*/
    do while (Lines(filePath,'c') > 0)
        /*Zeile aus Datei einlesen*/
        rc = linein(filePath)
        Select
            /* Lines Einlesen */
            when (word(rc,1) == 'Lines') then do
                arrayLines.LinesCounter = rc
                LinesCounter = LinesCounter + 1
            End
            /* Wort Einlesen */
            when (word(rc,1) == 'Words') then do
                arrayWords.WordsCounter = rc
                WordsCounter = WordsCounter + 1
            End
            otherwise do
            End
        End
    End
End
```

```
Say 'Read configuration (config.txt) succeeded.'  
End  
Else do  
    Say filePath' does not exist. Error Code: 'rc  
End  
i=0  
do while (i < LinesCounter)  
    Say arrayLines.i  
    i = i + 1  
End  
i=0  
do while (i < WordsCounter)  
    Say arrayWords.i  
    i = i + 1  
End  
Return
```

1.6.7.8.3 Logdatei erstellen

Logdatei erstellen

```
WarnCounter = 0  
ErrorCounter = 0  
logFile = ''  
/*Über diese Variable können die Debug Einträge Aktiviert oder  
Deaktiviert werden*/  
/* 1 .. aktiviert die DebugEinträge  
jeder andere Wert deaktiviert sie*/  
debug = 0  
/*Festplatte feststellen von welcher das Skript ausgeführt  
wird.*/  
Parse Source . . filename  
drive = Left(filename,2)  
drive = drive '\'
```



```
SetLogdataToFile : Procedure Expose logFile WarnCounter
ErrorCounter debug
Parse Arg LogString, Flag
actTime = TIME()
/*Set flag*/
stringFlag = ' '
Select
    When (Flag == 2) Then do
        stringFlag = '*config*'
    End
    When (Flag == 1) Then do
        stringFlag = '*warning*'
        WarnCounter = WarnCounter + 1
    End
    When (Flag == 0) Then do
        stringFlag = '*notice*'
    End
    When (Flag == 99) then do
        stringFlag = '*debug*'
    End
    Otherwise do
        stringFlag = '*error*'
        ErrorCounter = ErrorCounter + 1
    End
End
if ((debug == 1) | (Flag <> 99)) then do
    Say LogString
    retcode = LINEOUT(logFile, actTime' 'stringFlag'
'LogString)
    if (retcode <> 0) then do
```

1.6.7.8.4 Erstellen eines Pfads

Erstellen eines Pfads

```
make_dir: Procedure
  Parse Arg dir
  n = 0
  flag_End = 0
  do Forever
    /* strip trailing backslashes */
    revdir = reverse(dir)
    do Forever
      if (substr(revdir,1,1) == '\') Then do
        if (substr(revdir,2,1) \= '') &
        (substr(revdir,2,1) \= ':') Then do
          revdir = substr(revdir,2)
        End
        Else do
          flag_End = 1
        Leave
      End
    End
    Else do
      Leave
    End
  End
  dir = reverse(revdir)
  if (stream(dir,'C','QUERY EXISTS') \= '') | (dir =
  '') | (flag_End = 1) Then do
    /* create all other directories on the stack
   */
    do While (n > 0)
      Pull dir
      if (pos(' ', dir, 1) \= 0) Then
```

```

temp = ""dir""
else
    temp = dir
dir = temp
cmd = 'MD 'dir
Address System cmd
if (rc \= 0) Then do
    Say 'Creating Folder ('dir')
failed. Error Code: 'rc
Return rc
End
n = n - 1
End
Leave
End

```

```

push dir
n = n + 1
if (n > 50) Then do
    /* for safety */
    Say 'Too many Subfolders. Creating folder
structure ('dir') failed.'
Return -999
End
/* strip all characters until ':'', '\' is found or
string is empty */
revdir = reverse(dir)
do Forever
    c = substr(revdir,1,1)
    if (c = '\') | (c = ':') Then do
        Leave
    End

```

```
        revdir = substr(revdir,2)
    End
    dir = reverse(revdir)
End
retcode = MILLISLEEP(200)
counter = 0
do forever
    /*Only return 0 when the destination directory is
existing*/
    if (stream(dir,'C','QUERY EXISTS') == '') Then do
        if (counter > 30) then do
            call SetLogdataToFile 'The destination
directory ('dir') could not be created on the USB-Drive. Time
exceeded.', '-1'
            Return -998
        leave
    End
    counter = counter + 1
    retcode = MILLISLEEP(500)
End
Else do
    leave
End
End
Return 0
```

1.6.7.8.5 Kopieren eines Ordners 1

Kopieren von einem Ordner mit Unterordner und Inhalt auf einer lokalen CPU

```
copyTerminal: Procedure
    Parse Arg Source, Dest
    rc = 0
```

```

counter = 0
countError = 0
if (stream(Dest,'C','QUERY EXISTS') == '') then do
    rc = make_dir(Dest)
End
if (right(Dest,1) == '\') then do
    Dest = left(Dest, length(Dest) - 1)
End
if (right(Source,1) == '\') then do
    Source = left(Source, length(Source) - 1)
End
if (rc == 0) then do
    rc = Dirlist(Source'\*',list)
    if (rc == 0) Then do
        i = 1
        do while (i <= list.0)
            /*Creating command and paths*/
            if (list.i.A_SUBDIR == 0) Then do
                /*Gänsefüßchen vermutlich wegen
Dateinamen mit Leerzeichen*/
                if (pos(' ', Source, 1) \= 0)
Then do
                cpcmdsrc =
' "'Source'\list.i.NAME'"'
                End
                else do
                cpcmdsrc =
Source'\list.i.NAME
                End
                if (pos(' ', Dest, 1) \= 0) Then do

```

```
cpcmddst =
  Dest'\'list.i.NAME'
End
else do
  cpcmddst =
Dest'\'list.i.NAME
End
  if (stream(cpcmddst,'C','QUERY
EXISTS') <> '') then do
    /*Ändern der
Dateiattribute auf nicht Schreibgeschützt falls sie existieren,
damit man sie überschreiben kann*/
    Address System 'attrib
'cpcmddst' -r'
End
  say 'Copying 'cpcmddst' to
'cpcmddst
  cmd = 'COPY 'cpcmddst' 'cpcmddst
  Address System cmd
  if (rc <> 0) then do
    Say 'Copying 'cpcmddst'
failed. Error Code: 'rc
  End
End
Else do
  if ((list.i.NAME \= '..') &
(list.i.NAME \= '...')) Then do
    rc =
copyTerminal(Source'\'list.i.NAME, Dest'\'list.i.NAME)
  End
End
  i = i + 1
End
End
```

1.6.7.8.6 Kopieren eines Ordners 2

Kopieren von einem Ordner mit Unterordner und Inhalt von einer lokale CPU auf eine Remote-CPU

```
updateRemoteCPU: Procedure
  Parse Arg desthandle, Source, Dest
  if (right(Dest,1) == '\') then do
    Dest = left(Dest, length(Dest) - 1)
  End
  if (right(Source,1) == '\') then do
    Source = left(Source, length(Source) - 1)
  End
  rc = Dirlist(Source '\*',list)
  if (rc == 0) Then do
    i = 1
    do while (i <= list.0)
      /*Befehl und Pfade erstellen*/
      if (list.i.A_SUBDIR == 0) Then do
        /*Gänsefüßchen vermutlich wegen
        Dateinamen mit Leerzeichen*/
        if (pos(' ', Source, 1) \= 0) Then do
          cpcmdsrc =
            ""Source '\'' list.i.NAME """
        End
        else do
          cpcmdsrc = Source '\'' list.i.NAME
        End
        if (pos(' ', Dest, 1) \= 0) Then do
          cpcmddst =
            ""Dest '\'' list.i.NAME """
        End
        else do
          cpcmddst = Dest '\'' list.i.NAME
        End
      End
    End
  End
End
```

```
End
if (stream(cpcmddst, 'C', 'QUERY EXISTS')
<> '') then do
/*Change Attrib Writeprotected of
existing files at the destination to not write protected*/
Address System 'attrib 'cpcmddst'
-r'
End
Say 'Copying 'cpcmdsrc' to 'cpcmddst
rc =
TXFILE('DESCR', desthandle, cpcmdsrc, cpcmddst)
if (rc <> 0) then do
if (rc == '-307') then do
Say 'An Error on the
Remote-CPU occurred. Copying 'cpcmdsrc' failed.'
End
Else do
if (rc <> '') then do
Say 'Copying
cpcmdsrc' failed. Error Code: 'rc
End
Else do
Say 'Copying
cpcmdsrc' failed. The System does not support the command.
Update the OS first.'
End
End
End
Else do
if ((list.i.NAME \= '.') & (list.i.NAME
\= '..')) Then do
```

```

rc =
updateRemoteCPU
(desthandle,Source'\list.i.NAME,Dest'\list.i.NAME)
End
End
i = i + 1
End
End
Else do
    Say 'Creating the Dirlist of this Folder ('Source')
failed. Error Code: 'rc
End
return rc

```

1.6.7.8.7 Kopieren eines Ordners 3

Kopieren von einem Ordner mit Unterordner und Inhalt von einer Remote-CPU auf eine lokale CPU

```

backupRemoteCPU: Procedure Expose DirListName terminalCPUType
    Parse Arg desthandle, CPUType, Source, Dest
    if (right(Dest,1) == '\') then do
        Dest = left(Dest, length(Dest) - 1)
    End
    if (right(Source,1) == '\') then do
        Source = left(Source, length(Source) - 1)
    End
    /*Pfad zur Dirlist*/
    SourceDirListFile = Source'\DirListName
    /*Create a new DirList on the remote CPU*/
    /*Kommando zum erstellen einer DIRLIST auswählen damit es
    auch ausgeführt wird. Falls CPU-Type vom Terminal nicht bekannt
    ist wird der Befehl mit EXEC ausgeführt, was dem Befehl für ARM
    entspricht.*/

```

```
if (terminalCPUType == 'x86') then do
    cmd = 'DIR 'Source'\* > 'SourceDirListFile
End
Else do
    cmd = 'EXEC DIR 'Source'\* > 'SourceDirListFile
End
/*Dirlist erstellen*/
rc = REXEC('DESCR', desthandle, cmd)
ret = MILLISLEEP(1000)
if (rc == 0) then do
    /*Dirlist kopieren und destination ordner erstellen*/
    rc = make_dir(Dest)
    if (rc == 0) then do
        rc = RXFILE('DESCR', desthandle,
SourceDirListFile, Dest+'\DirListName')
    /*löschen des Dirlist-Files auf der Remote-CPU
da es kopiert wurde und nicht nocheinmal kopiert werden soll*/
        if (rc == 0) then do
            /*Kommando zum löschen einer DIRLIST
auswählen damit es auch ausgeführt wird. Falls CPU-Type vom
Terminal nicht bekannt ist wird der Befehl mit EXEC ausgeführt,
was dem Befehl für ARM entspricht.*/
            if (terminalCPUType == 'x86') then do
                cmd = 'DEL 'SourceDirListFile
            End
        Else do
            cmd = 'EXEC DEL
'SourceDirListFile
        End
    /*Deleting Dirlist on remote CPU*/
    rc = REXEC('DESCR', desthandle, cmd)
    ret = MILLISLEEP(1000)
```

```
if (rc == 0) then do
    /*Dirlist parsen und Dateien und
Unterordner kopieren*/
    do while
        (LINES(Dest'\DirListName))
        rc =
        LINEIN(Dest'\DirListName)
        if (CPUType == 'ARM') then
        do
            /*überprüfen ob
diese Zeile gültig ist. Sie ist gültig wenn sie 4 Wörter (Ordner
oder Datei namen mit Leerzeichen) hat und wenn das zweite Wort
nicht File(s) ist, da es eine Zeile gibt die auch noch 4 Wörter
hat*/
            if ((words(rc) == 4)
& (word(rc, 2) <> 'File(s)')) then do
                /*überprüfen
ob dieser PFAD kein Ordner ist, wenn ja wird der PFAD für die
Datei erstellt wenn nein dann wird diese Funktion für den
Unterordner nochmals aufgerufen*/
                if (word(rc,
3) <> '<DIR>') & (word(rc,4) <> DirListName)) then do
                    fileToCopy = word(rc,4)
                    if (rc
<> 0) then do
                        if (rc == '-307') then do
                            Say 'An Error on the Remote-CPU occurred. Copying file
('Source'\fileToCopy') failed.'
                    End
                Else do
                    if (rc <> '') then do
```

```
        Say 'Copying File ('Source'\'fileToCopy') failed.
Error code: 'rc

        End

        Else do

            Say 'Copying File ('Source'\'fileToCopy') failed. The
System does not support the command. Update the OS first.

        End

        End

        End

        Else do

            if
(word(rc,4) <> DirListName) then do

folderToCopy = word(rc,4)

Say 'Copying 'Source'\'folderToCopy

rc = backupRemoteCPU(desthandle, CPUType, Source'\'folderToCopy,
Dest'\'folderToCopy)

        End

        End

        End

        End

        Else do

            DirListNameLOW =
translate(terminalCPUType, lower, upper)

            /*überprüfen ob
diese Zeile gültig ist. Sie ist gültig wenn sie 5 Wörter hat*/
```

```
if (words(rc) == 5)
then do
    /*überprüfen
ob dieser PFAD kein Ordner ist, wenn ja wird der PFAD für die
Datei erstellt wenn nein dann wird diese Funktion für den
Unterordner nochmals aufgerufen*/
    if ((word(rc, 4) <> '<DIR>') & (word(rc,5) <> DirListName) & (word(rc,5) <>
DirListNameLOW)) then do
        fileToCopy = word(rc,5)
        Say
        'Copying 'Source'\fileToCopy
        rc =
RXFILE( 'DESCR' , desthandle , Source'\fileToCopy ,
Dest'\fileToCopy )
        if (rc
<> 0) then do
            Say 'Copying File ('Source'\fileToCopy') failed. Error code: 'rc
            End
        End
        Else do
/*Löschen der Dirlist.txt auf dem Stick*/
            cmd = 'DEL 'Dest'\DirListName'
            rc =
stream(Dest'\DirListName,'C','CLOSE')
            Address System cmd
            if (rc <> 0) then do
                Say 'Deleting the Dirlist.txt-
File ('Dest'\DirListName') on the Terminal failed. Error code:
'rc
            End
        End
    Else do
```

```
        Say 'Copying the Dirlist.txt-File
('SourceDirListFile') failed. Error code: 'rc
        End
    End
Else do
    if (rc == '-999') then do
        Say 'The destination directory could not
be created on the USB-Drive. Too many Subfolders.'
        return -999
    End
Else do
    if (rc == '-998') then do
        Say 'The destination directory
could not be created on the USB drive. Time exceeded.'
        return -998
    End
Else do
    Say 'The destination directory
could not be created on the USB drive. Error code: 'rc
    End
End
Else do
    Say 'Creating the Dirlist.txt file
('SourceDirListFile') failed. Error code: 'rc
    End
return rc
```

1.6.7.9 Notepad++ Highlighting

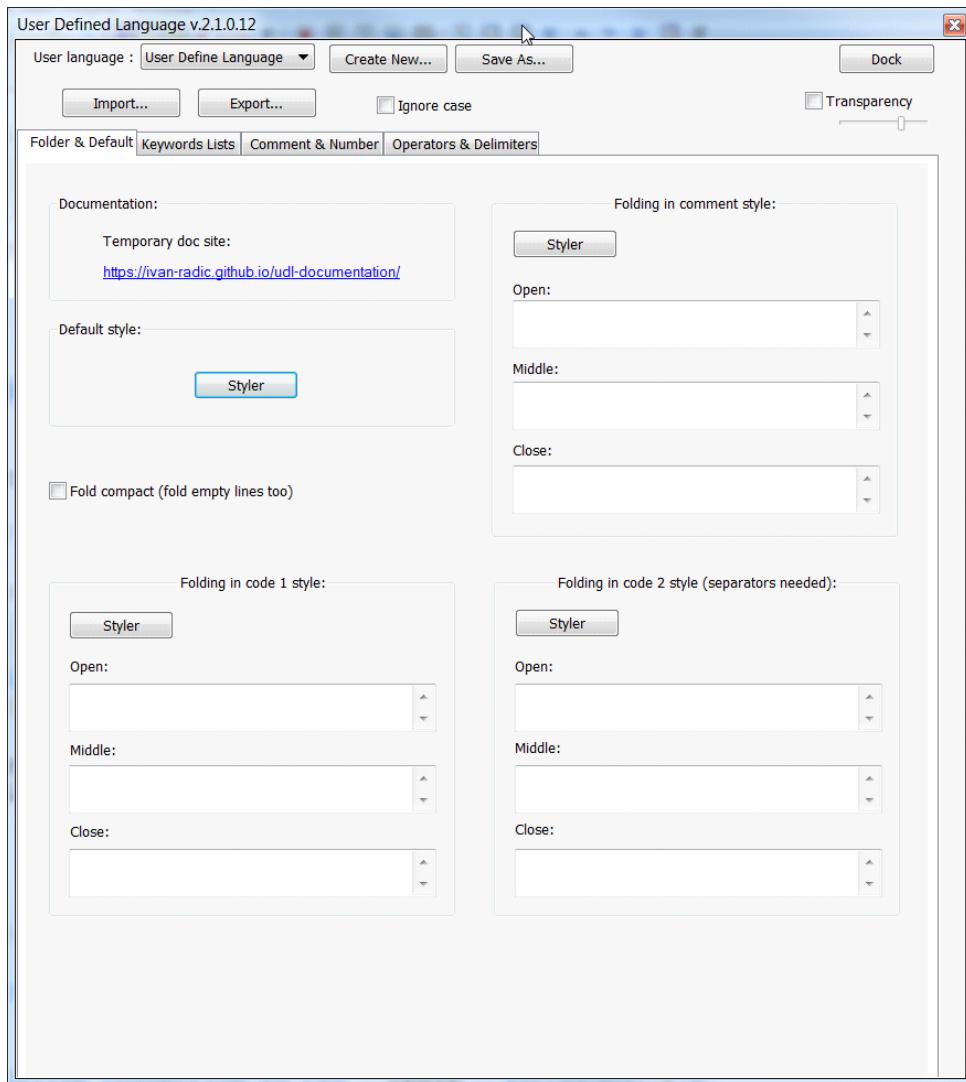
Zum Einfügen von Code Highlighting eine neue Textdatei erstellen, den Code aus [Anhang 1](#) hineinkopieren und die Datei als „REXX.xml“ benennen. Dann Notepad++ öffnen und dort in der Menüleiste „Language“ auswählen und auf „Define your Language...“ klicken. Dann öffnet sich ein Fenster, in diesem nun auf „Import“ klicken und die erstellte „REXX.xml“-Datei auswählen. Danach Notepad++ neustarten und nun kann im „Language“-Menü unter „Define your Language...“ „REXX“ ausgewählt werden.

File Edit Search View Encoding **Language** Settings Tools Macro

A
B
C
D
F
Gui4Cli
H
I
J
KIXtart
L
M
N
Objective-C
P
R
S
T
V
XML
YAML
Define your language...
REXX
User-Defined

AUTOSTRT.REX

```
5501
5502  /* -----
5503
5504
5505  /*-----
5506  #online_plc: I
5507
5508  /*get the
5509  Parse arg
5510
5511  call SetI
5512
5513  retrycour
5514  desthand]
5515
5516  /*online
5517  #if (desth
5518  /*try
5519  do wh
5520
5521          say 'Online connecti
```



1.6.7.10 Links

Ein paar nützliche Webseiten zur REXX-Programmierung

Kilowattsoftware

Auf dieser Webseite wird so ziemlich jede Standard-REXX-Funktion sehr gut erklärt, auch anhand von Beispielen. Man muss sich nur ein bisschen durchklicken.

<http://www.kilowattsoftware.com/tutorial/rexx/default.htm>

Rexxinfo

Hier findet man auch sehr viele verschiedene Funktionen erklärt. Weiters gibt es dort noch ein Verzeichnis von vielen anderen REXX-Tutorials.

<http://www.rexxinfo.org/html/functions.html> - REXX Funktionen

<http://www.rexxinfo.org/html/instructions.html> - REXX Anweisungen

<http://www.rexxinfo.org/html/rexxinfo1.html> - Verzeichnis mit REXX Tutorials

Tutorialspoint

Auch eine gute Webseite mit einem Tutorial für REXX. Diese Webseite bietet auch eine Online-REXX-Entwicklungsumgebung an. Sehr hilfreich um kleine Teile wie das Parsen von Strings oder ähnliches zu testen.

<https://www.tutorialspoint.com/rexx/index.htm> - tutorial

https://www.tutorialspoint.com/execute_rexx_online.php Entwicklungsumgebung erwartet werden

1.6.7.11 Code Highlighting

<NotepadPlus>

```
<UserLang name="REXX" ext="rex nrx" udlVersion="2.1">
  <Settings>
    <Global caseIgnored="yes" allowFoldOfComments="no"
foldCompact="no" forcePureLC="0" decimalSeparator="0" />
    <Prefix Keywords1="no" Keywords2="no" Keywords3="no"
Keywords4="no" Keywords5="no" Keywords6="no" Keywords7="no"
Keywords8="no" />
  </Settings>
```

```
<KeywordLists>
  <Keywords name="Comments">03/* 04*/ 00-- 01
02</Keywords>
  <Keywords name="Numbers, prefix1"></Keywords>
  <Keywords name="Numbers, prefix2"></Keywords>
  <Keywords name="Numbers, extras1"></Keywords>
  <Keywords name="Numbers, extras2"></Keywords>
  <Keywords name="Numbers, suffix1"></Keywords>
  <Keywords name="Numbers, suffix2"></Keywords>
  <Keywords name="Numbers, range"></Keywords>
  <Keywords name="Operators1">&lt; . [ \ ] { | } ~ + &lt; = &gt;</Keywords>
  <Keywords name="Operators2"></Keywords>
  <Keywords name="Folders in code1, open">DO PROCEDURE
::METHOD ::ROUTINE</Keywords>
  <Keywords name="Folders in code1, middle"></Keywords>
  <Keywords name="Folders in code1, close">END
RETURN</Keywords>
  <Keywords name="Folders in code2, open"></Keywords>
  <Keywords name="Folders in code2, middle"></Keywords>
  <Keywords name="Folders in code2, close"></Keywords>
  <Keywords name="Folders in comment, open"></Keywords>
  <Keywords name="Folders in comment, middle"></Keywords>
  <Keywords name="Folders in comment, close"></Keywords>
  <Keywords name="Keywords1">ADDRESS ARG BOOLEAN BYTE
CALL CLASS CROSSREF DIAG DOUBLE DROP EXIT EXPOSE FLOAT FORWARD
GUARD IMPORT INT INTERPRET ITERATE LEAVE LONG METHOD NOCROSSREF
NODIAG NOFORMAT NOP NOREPLACE NOSTRICTARGS NOSTRICTASSIGN
NOSTRICTCASE NOSTRICTSIGNAL NOTRACE NOUTF9 NOVERBOSE NOVERBOSEX
NULL NUMERIC OPTIONS PACKAGE PARSE PROPERTIES PULL PUSH QUEUE RAISE
REPLY REQUIRES REXX SAY SELECT SELF SHORT SIGNAL STRICTARGS
STRICTASSIGN STRICTCASE STRICTSIGNAL SUPER TRACE USE UTF9 VERBOSE
VERBOSEX</Keywords>
```

```
<Keywords name="Keywords2">::CLASS ::REQUIRES ABSTRACT
ADDITIONAL ALL ANY APPEND ARRAY BINARY BOTH BY CASELESS CATCH CHAR
CLOSE CONSTANT DATETIME DIGITS ELSE ENGINEERING ERROR EXISTS
EXTENDS FAILURE FINAL FINALLY FLUSH FOR FOREVER FORM FUZZ HALT
HANDLE IF IMPLEMENTS INHERIT INHERITABLE INTERFACE LABEL LINE
LOSTDIGITS LOWER NAME NATIVE NEW NOBUFFER NOMETHOD NORMAL NOSTRING
NOTREADY NOVALUE OBJECT OFF ON OPEN OTHERWISE OVER POSITION PRIVATE
PROTECT PUBLIC PUT QUERY READ RECLENGTH REPLACE RESULTS RETURNS
SCIENTIFIC SEEK SIGNALS SIZE SORT SORTADDITIONAL SOURCE STATIC
STREAMTYPE SUBCLASS SYNTAX SYS THEN TIMESTAMP TO UNTIL UPPER USER
USERID USES VALUE VAR VARIABLE VERSION VOLATILE WHEN WHILE WITH
WRITE</Keywords>
```

```
<Keywords name="Keywords3">ABBREV ABS BEEP BITAND BITOR
BITXOR B2X CENTER CENTRE ChangeStr CHARIN CHAROUT CHARS COMPARE
CONDITION COPIES COUNTSTR C2X DATE DATATYPE DELSTR DELWORD D2C D2X
DIRECTORY ERRORTEXT ENDLOCAL FILESPEC FORMAT INSERT LASTPOS LEFT
LENGTH LINEIN LINEOUT LINES MAX METHODS MIN OVERLAY POS QUEUED
RANDOM REVERSE RIGHT RS RxFuncAdd RxFuncDrop RxFuncQuery RxQueue
SetLocal SIGN SOURCELINE SPACE STREAM STRIP SUBSTR SUBWORD SYMBOL
TIME TRANSLATE TRUNC VERIFY WORD WORDINDEX WORDLENGTH WORDPOS WORDS
XRANGE X2B X2C X2D</Keywords>
```

```
<Keywords name="Keywords4">AbsRect2LogRect Add
AddAttribute AddAutoStartMethod AddBitmapButton AddBlackFrame
AddBlackRect AddButton AddButtonGroup AddCategoryComboEntry
AddCategoryListEntry AddCheckBox AddCheckBoxStem AddCheckGroup
AddComboBox AddComboEntry AddComboInput AddDirectory AddEntryLine
AddEtchedFrame AddEtchedHorizontal AddEtchedVertical AddFullSeq
AddGrayFrame AddGrayRect AddGroupBox AddInput AddInputGroup
AddInputStem AddListBox AddListControl AddListEntry AddMenuItem
AddMenuSeparator AddOkCancelLeftBottom AddOkCancelLeftTop
AddOkCancelRightBottom AddOkCancelRightTop AddPasswordLine
AddPopupMenu AddProgressBar AddRadioButton AddRadioGroup
AddRadioStem AddRow AddScrollBar AddSequence AddSliderControl
AddStyle AddTabControl AddText AddTreeControl AddUserMsg
AddWhiteFrame AddWhiteRect AdjustToRectangle AlignLeft AlignTop
Arrange AskDialog AssignFocus AssignWindow AsyncMessageHandling
AutoDetection BackgroundBitmap BackgroundColor BkColor Cancel
CaptureMouse CategoryComboAddDirectory CategoryComboDrop
CategoryListAddDirectory CategoryListDrop CategoryPage ChangeBitmap
```

ChangeBitmapButton ChangeCategoryComboEntry ChangeCategoryListEntry
ChangeComboEntry ChangeListEntry ChangePage Check CheckMenuItem
Child Clear ClearButtonRect ClearMessages ClearRect ClearSelRange
ClearTicks ClearWindowRect ClientToScreen CloseDropDown Collapse
CollapseAndReset ColumnInfo ColumnWidth CombineELwithSB
ComboAddDirectory ComboDrop ConnectAllSBEEvents
ConnectAnimatedButton ConnectBitmapButton ConnectButton
ConnectButtonNotify ConnectCheckBox ConnectComboBox
ConnectComboBoxNotify ConnectCommonNotify ConnectControl
ConnectDraw ConnectEditNotify ConnectEntryLine ConnectList
ConnectListBox ConnectListBoxify ConnectListControl
ConnectListLeftDoubleClick ConnectListNotify ConnectMenuItem
ConnectMouseCapture ConnectMove ConnectMultiListBox
ConnectPosChanged ConnectRadioButton ConnectResize ConnectScrollBar
ConnectScrollBarNotify ConnectSliderControl ConnectSliderNotify
ConnectStaticNotify ConnectTabNotify ConnectTreeControl
ConnectTreeNotify CountTicks Create CreateBrush
CreateCategoryDialog CreateCenter CreateFont CreateMenu CreatePen
CurrentCategory CursorPos Cursor_AppStarting Cursor_Arrow
Cursor_Cross Cursor_No Cursor_Wait DeInstall DeSelectIndex
DefListDragHandler DefTreeDragHandler DefineDialog Delete DeleteAll
DeleteCategoryComboEntry DeleteCategoryListEntry DeleteColumn
DeleteComboEntry DeleteFont DeleteListEntry DeleteObject Deselect
DeselectRange DeterminePosition DetermineSBPosition DimBitmap
Disable DisableCategoryItem DisableItem DisableMenuItem
DisplaceBitmap Display Draw DrawAngleArc DrawArc DrawBitmap
DrawButton DrawLine DrawPie DrawPixel DropHighlight DropHighlighted
Dump Edit EditSelection Enable EnableCategoryItem EnableItem
EnableMenuItem EndAsyncExecution EndEdit EnsureCaretVisibility
EnsureVisible ErrorDialog Execute ExecuteAsync Expand
FileNameDialog FillDrawing Find FindCategoryComboEntry
FindCategoryListEntry FindComboEntry FindListEntry FindNearestXY
FindPartial FirstVisible FirstVisibleLine Focus FocusCategoryItem
FocusItem Focused FontColor FontToDC ForegroundWindow FreeButtonDC
FreeDC FreeWindowDC Get GetArcDirection GetAttrib GetBitmapSizeX
GetBitmapSizeY GetBmpDisplacement GetButtonControl GetButtonDC
GetButtonRect GetCategoryAttrib GetCategoryCheckBox
GetCategoryComboEntry GetCategoryComboItems GetCategoryComboLine
GetCategoryEntryLine GetCategoryListEntry GetCategoryListItems
GetCategoryListLine GetCategoryMultiList GetCategoryRadioButton

GetCategoryValue GetCheckBox GetCheckControl GetClientRect
GetComboBox GetComboEntry GetComboItems GetComboLine
GetCurrentCategoryComboIndex GetCurrentCategoryListIndex
GetCurrentComboIndex GetCurrentListIndex GetDC GetData GetDataStem
GetEditControl GetEntryLine GetFirstVisible GetFocus GetID GetItem
GetLine GetLineStep GetListBox GetListControl GetListEntry
GetListItemHeight GetListItems GetListLine GetListWidth
GetMenuItemState GetMouseCapture GetMultiList GetPageStep GetPixel
GetPos GetProgressBar GetRadioButton GetRadioControl GetRect
GetSBPos GetSBRRange GetScrollBar GetSelectedPage GetSize
GetSliderControl GetStaticControl GetTabControl GetText GetTextSize
GetTick GetTreeControl GetValue GetWindowDC GetWindowRect
GrayMenuItem HScrollPos HandleMessages Help Hide HideCategoryItem
HideFast HideItem HideItemFast HideWindow HideWindowFast HitTest
Indent Indeterminate InfoDialog Init InitAutoDetection
InitCategories InitDialog InitRange InitSelRange
InsertCategoryComboEntry InsertCategoryListEntry InsertColumn
InsertComboEntry InsertListEntry IsAncestor IsChecked
IsDialogActive IsDropDownOpen IsModified IsMouseButtonDown
ItemHeight ItemInfo ItemPos ItemState ItemText ItemTitle Items
ItemsPerPage Last LastSelected Leaving LineFromIndex LineIndex
LineLength LineScroll ListAddDirectory ListDrop Load LoadBitmap
LoadFrame LoadItems LoadMenu LogRect2AbsRect MakeFirstVisible
Margins Maximize Minimize Modify ModifyColumn Move MoveCategoryItem
MoveItem MSSleep Next NextLeft NextPage NextRight NextSelected
NextVisible NoAutoDetection OK ObjectToDC OpaqueText OpenDropDown
PageHasChanged Parent PasswordChar PeekDialogMessage Play Popup
PopupAsChild PosRectangle Prepare4nItems Previous PreviousPage
PreviousSelected PreviousVisible ProcessMessage Range Rectangle
Redraw RedrawButton RedrawClient RedrawItems RedrawRect
RedrawWindow RedrawWindowRect ReleaseMouseCapture RemoveBitmap
RemoveImages RemoveSmallImages RemoveStyle ReplaceSelText
ReplaceStyle RequiredWindowSize Resize ResizeCategoryItem
ResizeItem RestoreCursorShape RestoreEditClass Root Rows Run
RxMessageBox RxWinExec ScreenSize ScreenToClient Scroll
ScrollBitmapFromTo ScrollButton ScrollCommand ScrollInButton
ScrollText SelRange SelectIndex SelectRange Selected SelectedIndex
SelectedIndexes SelectedItems SendMessageToCategoryItem
SendMessageToItem SetArcDirection SetAttrib SetCategoryAttrib
SetCategoryCheckBox SetCategoryComboLine SetCategoryEntryLine

SetCategoryItemFont SetCategoryListLine SetCategoryListTabulators
SetCategoryMultiList SetCategoryRadioButton SetCategoryStaticText
SetCategoryValue SetCheckBox SetColor SetColumnWidth SetComboLine
SetCurrentCategoryComboIndex SetCurrentCategoryListIndex
SetCurrentComboIndex SetCurrentListIndex SetCursorPos SetData
SetDataStem SetEntryLine SetFocus SetFont SetHScrollPos SetImages
SetItemFont SetItemPos SetItemState SetItemText SetLimit
SetLineStep SetListColumnWidth SetListItemHeight SetListLine
SetListTabulators SetListWidth SetMargins SetMax SetMenu
SetMenuItemRadio SetMin SetModified SetMultiList SetPadding
SetPageStep SetPos SetRadioButton SetRange SetReadOnly SetRect
SetsSBPos SetsSBRRange SetSelEnd SetSelStart SetSize SetSmallImages
SetStaticText SetStep SetTabulators SetTickAt SetTickFrequency
SetTitle SetVScrollPos SetValue SetWidth SetWindowRect
SetWindowTitle Show ShowCategoryItem ShowFast ShowItem ShowItemFast
ShowWindow ShowWindowFast SmallSpacing SnapToGrid SortChildren
Spacing StartIt State Step StopIt StringWidth Style SubclassEdit
SysAddFileHandle SysAddRexxMacro SysBootDrive
SysClearRexxMacroSpace SysCloseEventSem SysCloseMutexSem SysCls
SysCopyObject SysCreateEventSem SysCreateMutexSem SysCreateObject
SysCreatePipe SysCurPos SysCurState SysDriveInfo SysDriveMap
SysDropFuncs SysDropRexxMacro SysDumpVariables SysFileCopy
SysFileDelete SysFileExist SysFileMove SysFileSearch
SysFileSystemType SysFileTree SysFork SysFromUnicode SysGetCollate
SysGetErrortext SysGetFileDialogTime SysGetKey SysGetMessage
SysGetMessageX SysIni SysIsFile SysIsFileCompressed
SysIsFileDialog SysIsFileEncrypted SysIsFileLink
SysIsFileNotContentIndexed SysIsFileOffline SysIsFileSparse
SysIsFileTemporary SysLoadFuncs SysLoadRexxMacroSpace SysMkDir
SysOpenEventSem SysOpenMutexSem SysPostEventSem SysProcessType
SysPulseEventSem SysQueryProcess SysQueryProcessCodePage
SysQueryRexxMacro SysReleaseMutexSem SysReorderRexxMacro
SysRequestMutexSem SysResetEventSem SysRmDir SysSaveRexxMacroSpace
SysSearchPath SysSetFileDialogTime SysSetPriority
SysSetProcessCodePage SysShutdownSystem SysSleep SysStemCopy
SysStemDelete SysStemInsert SysStemSort SysSwitchSession
SysSystemDirectory SysTempFileName SysTextScreenRead
SysTextScreenSize SysToUnicode SysUtilVersion SysVersion
SysVolumeLabel SysWait SysWaitEventSem SysWaitNamedPipe SysWildCard
SysWinDecryptFile SysWinEncryptFile SysWinGetDefaultPrinter

```
SysWinGetPrinters SysWinSetDefaultPrinter SysWinVer TextBkColor
TextColor TiledBackgroundBitmap Title ToTheTop Toggle
TransparentText Uncheck UncheckMenuItem Update UpdateItem
VScrollPos Validate Value VisibleItems Width WriteDirect
WriteToButton WriteToWindow</Keywords>
    <Keywords name="Keywords5"></Keywords>
    <Keywords name="Keywords6"></Keywords>
    <Keywords name="Keywords7"></Keywords>
    <Keywords name="Keywords8"></Keywords>
    <Keywords name="Delimiters">00' 01 02'
03" 04 05" 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20
21 22 23</Keywords>
</KeywordLists>
<Styles>
    <WordsStyle name="DEFAULT" fgColor="000000"
bgColor="FFFFFF" fontName="" fontStyle="0" nesting="0" />
    <WordsStyle name="COMMENTS" fgColor="808080"
bgColor="FFFFFF" fontName="" fontStyle="0" nesting="0" />
    <WordsStyle name="LINE COMMENTS" fgColor="808080"
bgColor="FFFFFF" fontName="" fontStyle="2" nesting="0" />
    <WordsStyle name="NUMBERS" fgColor="800040"
bgColor="FFFFFF" fontName="" fontStyle="0" nesting="0" />
    <WordsStyle name="KEYWORDS1" fgColor="0000FF"
bgColor="FFFFFF" fontName="" fontStyle="0" nesting="0" />
    <WordsStyle name="KEYWORDS2" fgColor="008000"
bgColor="FFFFFF" fontName="" fontStyle="0" nesting="0" />
    <WordsStyle name="KEYWORDS3" fgColor="FF00FF"
bgColor="FFFFFF" fontName="" fontStyle="0" nesting="0" />
    <WordsStyle name="KEYWORDS4" fgColor="8000FF"
bgColor="FFFFFF" fontName="" fontStyle="0" nesting="0" />
    <WordsStyle name="KEYWORDS5" fgColor="000000"
bgColor="FFFFFF" fontName="" fontStyle="0" nesting="0" />
    <WordsStyle name="KEYWORDS6" fgColor="000000"
bgColor="FFFFFF" fontName="" fontStyle="0" nesting="0" />
```

```
        <WordsStyle name="KEYWORDS7" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="KEYWORDS8" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="OPERATORS" fgColor="FF8000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="FOLDER IN CODE1" fgColor="008000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="FOLDER IN CODE2" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="FOLDER IN COMMENT" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="DELIMITERS1" fgColor="0080FF"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="DELIMITERS2" fgColor="0080C0"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="DELIMITERS3" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="DELIMITERS4" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="DELIMITERS5" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="DELIMITERS6" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="DELIMITERS7" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
        <WordsStyle name="DELIMITERS8" fgColor="000000"  
        bgColor="#FFFFFF" fontName="" fontStyle="0" nesting="0" />  
    </Styles>  
  </UserLang>  
</NotepadPlus>
```

1.7 VNCcli

1.7.1 Was ist LasalOS VNCcli?

LasalOS VNCcli ist eine VNC-Client Applikation. Das VNC-Protokoll ist ein einfaches Protokoll für Remote-Zugriff auf grafischen Benutzerschnittstellen und ist auf dem remote framebuffer (RfB) Konzept basiert. Das Protokoll ermöglicht es einen Server dem Frame-Buffer einfach zu aktualisieren, der über einen Viewer angezeigt wird. Weil es auf der Frame-Buffer-Ebene funktioniert, kann sie mit allen Betriebs- oder Windows-Systemen und Anwendungen benutzt werden.

Der VNC-Client ermöglicht dem Anwender, in jedem beliebigen Remote-System zu arbeiten, das einen VNC-Server (Lasal OS, Windows, MacOS, Linux, Unix, ...) betreibt.

VNC-Server für Windows, Linux, Unix und anderen Systemen sind auf den folgenden Seiten kostenlos erhältlich: <http://www.realvnc.com> und <http://www.tightvnc.com>.

1.7.2 Der Zweck dieses Dokuments

Dieses Dokument beschreibt die LasalOS-Schnittstelle für den VNC-Client Applikation.

Anforderungen

LasalOS	Benötigt LasalOS 1.1.3
Platform	LasalOS VNCcli ist auf der C-IPC vorhanden

Einschränkungen

Die ursprüngliche Applikation unterstützt verschiedene Farbmodi, einschließlich indizierter Paletten. Da das original Pixelformat für die C-IPC ist 16-Bit-rgb565 (2 Byte pro Pixel, 5 Bit Rot, 6 Bit Grün, 5 Bit Blau), ist es derzeit das einzige unterstützte Pixelformat. Weil alle VNC-Server kann jedes Pixelformat zur Verfügung stellen, sollte dies kein Problem werden.

Ab Salamander 09.04.010

Salamander 09.02.150

Gecko 09.07.040

werden nun die Farbtiefe 8 Bit, 16 Bit und 32 Bit unterstützt, davor nur 16 Bit.

Einbauzustand

- Um den VNC-Client mit einem C-IPC verwenden zu können, kopieren Sie die VNCCLI.DLM und ZLIB.DLM Dateien in das Verzeichnis C:\LSLSYS des Zielsystems.
- Bearbeiten Sie die AUTOEXEC.LSL und fügen Sie die Zeile "VNCCLI START" ein, um den VNC-Client Service beim Hochfahren zu starten.

1.7.3 LASAL OS VNCcli-API Funktionen

Die VNCCLI OS-Schnittstelle wird verfügbar, wenn der VNC-Client die Anwenderplattform unterstützt und der VNC-Client Dienst gestartet wird, indem Sie den Befehl START VNCCLI in der Datei AUTOEXEC.LSL eingefügen. Die Schnittstellen-Struktur, die Funktions-Prototypen und die Konstanten sind in der lsl_st_vnc.h Datei definiert.

Funktionen mit einem DINT als Rückgabeparameter sind fehlgeschlagen, wenn sie einen Wert größer als 0 sein zurückliefern. Sehen Sie [Error Codes](#) für eine Beschreibung den Fehler-Codes.

1.7.3.1 Die OS-Schnittstelle VNCCLI

```
TYPE
  OS_VNCCLI           : STRUCT
    udSize            : UDINT;
    udVersion         : UDINT;
    udVersionVNC     : UDINT;
    VNCCInit          : pvoid;
    VNCExit           : pvoid;
    VNCCConnect        : pvoid;
    VNCDDisconnect    : pvoid;
    VNCDDisconnectHandle : pvoid;
    VNCGetSize         : pvoid;
    VNCReset           : pvoid;
    VNCSendKbrdEvent   : pvoid;
    VNCSendHIDEEvent   : pvoid;
    VNCDraw             : pvoid;
    VNCForceRepaint    : pvoid;
    VNCSendKeyboardEvents : pvoid;
    VNCSendPointerEvents : pvoid;
    VNCSetClisleep     : pvoid;
    VNCFreeBuffer       : pvoid;
    VNCSetTaskDelayTime : pvoid;
    VNCFreeOldSockets   : pvoid;
    VNCDraw2            : pvoid;
    VNCSetKaValues      : pvoid;
    VNCSetConnectTimeout : pvoid;
    VNCSetOptions        : pvoid;
  END_STRUCT;
```

END_TYPE

Das Folgende ist ein kurzes Beispiel dafür, wie diese Schnittstelle verwendet wird:

Beispiel

```
...
VAR
  pVNCCLI : ^OS_VNCCLI;
  retv    : SYS_ERROR;
END_VAR;

  retv := OS_CILGet("VNCCLI", #pVNCCLI$void);

  if retv <> SYS_ERR_NONE then
    pVNCCLI := 0$^OS_VNCCLI;
    TRACE("No VNC-OS-Interface! Check your OS-Version-Number!");
  else
    err := pVNCCLI^.VNCExit $ G_VNCExit();
  end_if;

  OS_CILRelease("VNCCLI");
...

```

1.7.3.2 Callback-Funktion

Der VNC-Client-Service muss in der Lage sein, Änderungen im Verbindungsstatus dem entsprechenden LASAL CLASS-Objekt mitzuteilen. Wenn der VNC-Server z.B. die Verbindung deaktiviert, muss das LASAL-Objekt über der Änderung informiert werden. Eine globale Callback-Funktion muss daher registriert werden, die aufgerufen wird, wenn ein Objekt von LASAL CLASS über eine Statusänderung benachrichtigt werden muss.

Für jede VNC-Verbindung kann der pThis Zeiger einer Anwender-Klasse als Parameter myThis der VNCConnect-Funktion übergeben werden. Dieser Zeiger sollte auf die Anwendungsklasse vncclient im LASAL CLASS Projekt zeigen. Der pThis Parameter der Callback-Funktion ist derselbe Wert wie myThis. Er kann einem Zeiger auf die Anwenderklasse zugewiesen werden. Dadurch kann die Callback-Funktion die Funktionen der Anwenderklasse aufrufen und auf die Member-Variablen der Anwenderklasse zugreifen. Siehe das Beispiel weiter unten.

Übergabeparameter	Typ	Beschreibung
pThis	PVOID	Dieser Parameter ist ein Zeiger auf die LASAL CLASS Objekt, das über Änderungen benachrichtigt werden soll. Dies ist derselbe Pointer/Wert, der als Parameter myThis der vncconnect-Funktion verwendet wird.
func	UDINT	Dieser Parameter ist ein Funktions-Code.

		Symbolname	Wert	Beschreibung												
		VNCMSG_CO NNECT	1	Jedes Mal verwendet, wenn eine Verbindung zu einem VNC-Server mit einem Aufruf der Funktion <code>VNCConnect</code> erstellt wird.												
		VNCMSG_DIS CONNECT	2	Dieser Funktions-Code benachrichtigt das LASAL-Objekt, wenn eine Verbindung beendet wird. Es wird normalerweise beim Aufruf von <code>VNCDisconnect()</code> ausgelöst, aber es wird auch ausgeführt, wenn der Server die Verbindung beendet hat oder ein Fehler aufgetreten ist.												
		VNCMSG_UP DATE	3	Dieser Funktions-Code wird verwendet, um die VNC-Nachricht zu aktualisieren.												
param	UDINT	Für jeden Funktions-Code (Parameter func) wird einer oder mehrere Parameter-Codes definiert:														
VNCMSG_CONNECT																
<table border="1"> <tr> <td>VNCMSG_CO NNECT_STAR TED</td><td>0</td><td>Der Verbindungsaufbau ist gestartet, aber eine Verbindung ist noch nicht erstellt geworden.</td></tr> <tr> <td>VNCMSG_CO NNECT_DON E</td><td>1</td><td>Eine Verbindung zu dem angegebenen Host wurde erstellt. Events können nun gesendet und der Remote-Bildschirm angezeigt werden.</td></tr> <tr> <td>VNCMSG_CO NNECT_FAILE D</td><td>2</td><td>Die Verbindung zum Host ist fehlgeschlagen. Das <code>_VNCHANDLE</code> wird jetzt ungültig.</td></tr> <tr> <td>VNCMSG_CO NNECT_AUTH FAILED</td><td>3</td><td>Die Verbindung zum Host war nicht erfolgreich; Authentifizierung aufgrund einer ungültigen Benutzer/Passwort-Kombination fehlgeschlagen; <code>_VNCHANDLE</code> ist jetzt ungültig.</td></tr> </table>					VNCMSG_CO NNECT_STAR TED	0	Der Verbindungsaufbau ist gestartet, aber eine Verbindung ist noch nicht erstellt geworden.	VNCMSG_CO NNECT_DON E	1	Eine Verbindung zu dem angegebenen Host wurde erstellt. Events können nun gesendet und der Remote-Bildschirm angezeigt werden.	VNCMSG_CO NNECT_FAILE D	2	Die Verbindung zum Host ist fehlgeschlagen. Das <code>_VNCHANDLE</code> wird jetzt ungültig.	VNCMSG_CO NNECT_AUTH FAILED	3	Die Verbindung zum Host war nicht erfolgreich; Authentifizierung aufgrund einer ungültigen Benutzer/Passwort-Kombination fehlgeschlagen; <code>_VNCHANDLE</code> ist jetzt ungültig.
VNCMSG_CO NNECT_STAR TED	0	Der Verbindungsaufbau ist gestartet, aber eine Verbindung ist noch nicht erstellt geworden.														
VNCMSG_CO NNECT_DON E	1	Eine Verbindung zu dem angegebenen Host wurde erstellt. Events können nun gesendet und der Remote-Bildschirm angezeigt werden.														
VNCMSG_CO NNECT_FAILE D	2	Die Verbindung zum Host ist fehlgeschlagen. Das <code>_VNCHANDLE</code> wird jetzt ungültig.														
VNCMSG_CO NNECT_AUTH FAILED	3	Die Verbindung zum Host war nicht erfolgreich; Authentifizierung aufgrund einer ungültigen Benutzer/Passwort-Kombination fehlgeschlagen; <code>_VNCHANDLE</code> ist jetzt ungültig.														
VNCMSG_DISCONNECT																
<table border="1"> <tr> <td>VNCMSG_DIS CONNECT_ST ARTED</td><td>0</td><td>Das Entkopplungsverfahren wird gestartet.</td></tr> </table>					VNCMSG_DIS CONNECT_ST ARTED	0	Das Entkopplungsverfahren wird gestartet.									
VNCMSG_DIS CONNECT_ST ARTED	0	Das Entkopplungsverfahren wird gestartet.														

		VNCMSG_DIS_CONNECT_DONE	1	Das Entkopplungsverfahren ist abgeschlossen. Das _VNCHandle ist jetzt ungültig.
		VNCMSG_DIS_CONNECT_EXIT	2	Der Client wurde abgekoppelt, weil der VNC-Service Client gestoppt wurde (VNCExit oder VNCReset wurde aufgerufen). Das _VNCHandle ist jetzt ungültig.
VNCMSG_UPDATE				
		VNCMSG_UPDATE_SCREEN	0	Der Remote-Bildschirm hat sich verändert und muss mit einem Aufruf von VNCDraw aktualisiert werden.

Prototyp

```
FUNCTION __CDECL GLOBAL MyVNCCallback
VAR_INPUT
    pThis    : pvoid;
    func     : UDINT;
    param    : UDINT;
END_VAR;
```

Umsetzungsbeispiel

Ist _VNCClient die LASAL-Klasse, die den VNC-Client implementiert:

```
FUNCTION __CDECL GLOBAL MyVNCCallback
VAR_INPUT
    pThis    : pvoid;
    func     : UDINT;
    param    : UDINT;
END_VAR
VAR
    pClient  : ^_VNCClient;
END_VAR
    if (pThis = 0) then
        return;
    end_if;
    pClient  := pThis$^_VNCClient;
    (* Hier wird den Member-Funktion der LASAL-Klasse aufgerufen.*)
    pClient^.VNCCallback(func, param);
END_FUNCTION //VIRTUAL GLOBAL _VncClient::VNCCallback
```

Die Member-Funktion VNCCallback der Klasse _VncClient kann wie in folgendem Beispiel implementiert werden:

Beispiel für die VNC-Callback-Funktion

```
FUNCTION GLOBAL _VncClient::VNCCallback
VAR_INPUT
    func      : UDINT;
    param     : UDINT;
END_VAR

case func OF
    VNCMSG_CONNECT :
        case param of
            VNCMSG_CONNECT_STARTED : ...
            ...
            ...
END_FUNCTION
```

1.7.3.3 udSize

KEIN FUNKTIONS-ZEIGER!

Eine UDINT Variable, die die Größe der VNC-Client Schnittstellen-Struktur enthält.

1.7.3.4 udVersion

KEIN FUNKTIONS-ZEIGER!

Eine UDINT Variable, die die DLL-Version der VNC-Client DLL enthält.

Derzeit wird nur das LOW-WORD verwendet, um die dreiteilige DLL-Versionsnummer zu definieren (#1.#2.#3):

HIGH WORD				LOW WORD			
31-28	27-24	23-20	19-13	15-12	11-8	7-4	3-0
Nicht verwendet (0)				#1	#2	#3	

Beispiel

DLL-Version 1.1.1 entspricht den Hex-Wert 16#00001101.

1.7.3.5 udVersionVNC

KEIN FUNKTIONS-ZEIGER!

Eine UDINT Variable, die die Versionsnummer des VNC-Protokolls enthält.

Das obere Wort enthält die Haupt-Versionsnummer, das unteren Wort enthält die Unter-Versionsnummer.

Beispiel

VNC-Protokoll Version 3.8 entspricht dem Hex-Wert 16#00030008.

1.7.3.6 VNCHandle

Initialisiert den VNC-Client. Diese Funktion muss vor allen anderen Funktionen dieser Schnittstelle aufgerufen werden.



Es ist wichtig einen Zeiger auf eine Callback-Funktion anzugeben, da LasalOS jene LASAL-Objekte, die zur Anzeige des Remote-Bildes verwendet werden, über diese Callback-Funktion benachrichtigen wird. Einen NIL-Zeigers anzugeben bedeutet, dass die VNC-Verbindung nie aufgebaut werden kann!

Übergabeparameter	Typ	Beschreibung
logStr	^CHAR	<p>Dieser Parameter ist ein String, der eine Log-Einstellung definiert. Der Log-String ist in drei Abschnitte unterteilt, die durch Doppelpunkte getrennt sind: „[filter]:[stream]:[log level]“. Der erste Abschnitt definiert einen Modulnamen-Filter „*“ und protokolliert Nachrichten aus allen Modulen; der zweite Abschnitt definiert einen Output-Stream. Der Stream-Name „file“, schreibt die Log-Nachrichten in eine Datei (der Dateiname ist mit dem zweiten Parameter dieser Funktion definiert). Andere Stream-Namen könnten aus ‚stderr‘, ‚stdout‘ oder ‚Kernel‘ bestehen. Im dritten Abschnitt werden die Log-Level definiert (0: nur Informationsnachrichten und Fehler werden protokolliert, 200: alle Nachrichten einschließlich Debug-Nachrichten werden protokolliert). Ein wesentlicher logStr wäre: “*:kernel:30”.</p> <div style="text-align: center;">  <p>Verwenden Sie ‚Kernel‘ als einen Log-Stream, um die VNC Io-Nachrichten in die System-Log-Datei C:\sysmsg\event00.log zu schreiben.</p> </div>

LogFileName	^CHAR	Ist "file" im logStr Parameter als ein Output-Stream definiert, muss ein Namen für die Log-Datei zur Verfügung gestellt werden oder dieser Parameter wird ignoriert.
PVNCCallback	PVOID	Das Betriebssystem muss in der Lage sein, Änderungen in einer VNC-Session (Connect, Disconnect, neu, ...) einem LASAL-Objekt mitzuteilen; die globale Callback-Funktion muss angegeben werden.

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCInit
VAR_INPUT
    logStr      : ^CHAR;
    logFileName : ^CHAR;
    Callback    : pvoid;
END_VAR;
```

Log Levels

Log level	Nachrichtentyp	Beschreibung
0	fehler	Schreibt nur Fehler in die Protokolldatei.
30	Info	Schreibt zusätzliche Informationen zu den Protokollen.
100	Debug	Nur für das Debugging nötig.
150	Erweitert	Nur für das Debugging nötig.

1.7.3.7 VNCConnect

Diese Funktion wird zur Erstellung einer Verbindung mit dem angegebenen VNC-Server verwendet.

Übergabeparameter	Typ	Beschreibung
hostinfo	^CHAR	Dieser Parameter enthält den Host-Namen des Servers. Eine Port-Nummer kann auch nach dem Host-Namen hinzugefügt werden (nach einem Doppelpunkt, z.B. 10.10.100.100:2300), falls der VNC-Server nicht den Standard VNC-Port 5900 verwendet.
username	^CHAR	Dieser Parameter stellt den für die Authentifizierung benötigten Benutzernamen zur Verfügung. Benötigt der Server keinen Benutzernamen, muss der Parameter auf NIL gesetzt werden.
passwort	^CHAR	Der Server braucht das Passwort zur Authentifizierung. Benötigt der Server keinen Benutzernamen, muss auch der Parameter

		password auf NIL gesetzt werden.
myThis	pVoid	Dieser Parameter ist der THIS-Zeiger des LASAL CLASS Objekts, das die VNC-Verbindung verwaltet. Siehe auch: Beschreibung der Callback-Funktion.
prio	UDINT	Zum Einstellen der Priorität des VNC-Client-Thread. Zulässig sind Werte von 1 bis 13. Die Voreinstellung ist 5. Der Parameter wird nur beim Betriebssystem RTK verwendet.
dontfree	UDINT	Zulässige Werte sind 0 oder 1. Wert=1: Gibt den Speicher nach einem Disconnect nicht mehr frei und versucht beim nächsten Connect den Speicher wieder zu verwenden. Der Parameter wird nur beim Betriebssystem RTK verwendet.
Rückgabeparameter	Typ	Beschreibung
retval	_VNCHANDLE	Liefert ein _VNCHANDLE zurück, das die Verbindung identifiziert. Bei einem Fehler ist der Wert des Handles 0.



Die Funktion liefert sofort einen Wert zurück, jedoch bedeutet ein Verbindungs-Handle nicht, dass tatsächlich eine Verbindung hergestellt wurde. Jeder Funktionsaufruf ist asynchron. Andernfalls würde das System blockiert werden, bis die nächste Verbindung aufgebaut wurde oder ein Timeout auftritt, weil der Dienst nicht verfügbar ist..

Die Callback-Funktion liefert die Information, ob die Verbindung erfolgreich aufgebaut werden konnte oder nicht.

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCConnect
VAR_INPUT
    hostinfo      : ^CHAR;
    username      : ^CHAR;
    passwort      : ^CHAR;
    myThis        : pVoid;
    prio          : UDINT;
    dontfree      : UDINT;
END_VAR
VAR_OUTPUT
    retval        : _VNCHANDLE;
END_VAR;
```

1.7.3.8 **VNCDisconnect**

Diese Funktion beendet eine Verbindung mit einem angegebenen VNC-Server.

Übergabeparameter	Typ	Beschreibung
hostinfo	^CHAR	<p>Server-Name</p>  <p>Der angegebene Name muss genau der gleiche sein, wie der String, der zur Initialisierung der Verbindung mit der Funktion VNCConnect() verwendet wurde (inkl. Port-Angabe).</p>
Rückgabeparameter	Typ	Beschreibung
retval	_VNCHANDLE	<p>Betriebssystem RTK: Liefert den _VNCHANDLE der getrennten Verbindung. Wenn ein ungültiger String angegeben wurde oder die Trennung fehlgeschlagen ist, liefert die Funktion 0. Wenn die Funktion mit "all" als Parameter aufgerufen wurde, wird die Anzahl der getrennten Hosts zurückgegeben.</p> <p>Andere Betriebssysteme: immer 0.</p>

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCDisconnect
VAR_INPUT
    hostinfo      : ^CHAR
END_VAR
VAR_OUTPUT
    retval       : _VNCHANDLE;
END_VAR;
```

1.7.3.9 **VNCDisconnectHandle**

Diese Funktion beendet eine Verbindung zu einem angegebenen VNC-Server mittels dessen _VNCHANDLE Funktion.

Übergabeparameter	Typ	Beschreibung
hVNC	_VNCHANDLE	Von vncconnect zurückgegebenes Verbindungs-Handle
Rückgabeparameter	Typ	Beschreibung

retval	DINT	VNCHAN DLE	Funktion erfolgreich
			0 Ungültiger Handle angegeben oder Funktion fehlgeschlagen

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCDisconnectHandle
VAR_INPUT
    hVNC      : _VNCHANDLE;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;
```

1.7.3.10 VNCDraw

Diese Funktion wird zum Anzeigen des Remote-Bildes verwendet und wird nur in der Draw-Funktion der LASAL CLASS-Funktion aufgerufen (z.B. eine abgeleitete `_MyIO` Klasse aus der Library System-Visualisation-LseRtk-UserInterface). Siehe auch `VNCDraw2()`.

Übergabeparameter	Typ	Beschreibung	
hVNC	_VNCHANDLE	Von vncconnect zurückgegebenes Verbindungs-Handle	
clientRect	_ROOM	Client-Rechteck, enthält die absoluten Bildschirmkoordinaten, die zum Zeichnen des Remote-Bildes verwendet werden	
scroll_x	DINT	Bild links/rechts verschieben	
scroll_y	DINT	Bild rauh/runter verschieben	
Rückgabeparameter	Typ	Beschreibung	
retVal	DINT	0	Funktion erfolgreich
		#0	Ungültiges Handle angegeben oder Funktion fehlgeschlagen

`scroll_x` und `scroll_y` sind nur gültig, wenn das Client-Rechteck (`clientRect`) zu klein ist, um das vollständige Remote-Bild anzuzeigen.

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCDraw
```

```

VAR_INPUT
    hVNC      : _VNCHANDLE;
    clientRect : ^_ROOM;
    scroll_x  : DINT;
    scroll_y  : DINT;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;

```

1.7.3.11 VNCDraw2

Diese Funktion steht beim Betriebssystem RTK nicht zur Verfügung.

Eine schnelle Zeichenroutine, die ermittelt was neu gezeichnet werden muss und die entsprechenden Bereiche in den Bildpuffer kopiert.

Übergabeparameter		Typ	Beschreibung
_hVNC		_VNCHANDLE	Von vncconnect zurückgegebenes Verbindungs-Handle
clientRect		^_ROOM	Zeiger auf den Bereich am Bildschirm, in den gezeichnet werden soll (absolute Positionen bzgl. Screen)
scroll_x		DINT	Bild links/rechts verschieben (in Pixel)
scroll_y		DINT	Bild rauf/runter verschieben (in Pixel)
Rückgabeparameter		Typ	Beschreibung
retval		DINT	0 Funktion erfolgreich 1 Ungültiges Handle oder Funktion aufgrund eines anderen Fehlers fehlgeschlagen

scroll_x und scroll_y sind nur gültig, wenn das Client-Rechteck (clientRect) zu klein ist, um das vollständige Remote-Bild anzuzeigen.

Prototyp

```

FUNCTION __CDECL GLOBAL G_ VNCDraw2
VAR_INPUT
    _hVNC      : _VNCHANDLE;
    clientRect : ^_ROOM;
    scroll_x  : DINT;
    scroll_y  : DINT;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;

```

1.7.3.12 VNCExit

Diese Funktion trennt alle Clients und löscht die in [VNCInit\(\)](#) angegebene globale Callback-Funktion. Die [VNCInit\(\)](#) Funktion muss wieder aufgerufen werden, um den VNC-Client Service nach dem Aufruf von [VNCExit\(\)](#) zu verwenden.

Um alle Clients ohne Löschen der Callback-Information zu trennen, kann die [VNCDisconnect\("all"\)](#)-Funktion verwendet werden.



Parameter

Keine.

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCExit;
```

1.7.3.13 VNCForceRepaint

Der Aufruf dieser Funktion löst eine Aktualisierung des gesamten Client-Bereichs aus.

Übergabeparameter		Typ	Beschreibung	
<code>_hVNC</code>		<code>_VNCHANDLE</code>	Von vncconnect zurückgegebenes Verbindungs-Handle	
Rückgabeparameter		Typ	Beschreibung	
<code>retval</code>		<code>DINT</code>	0 Funktion erfolgreich ≠0 Ungültiges Handle angegeben oder Funktion fehlgeschlagen	

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCForceRepaint
VAR_INPUT
  _hVNC      : _VNCHANDLE;
END_VAR
VAR_OUTPUT
  retval     : DINT;
END_VAR;
```

1.7.3.14 VNCFreeBuffer

Diese Funktion ist nicht implementiert.

Übergabeparameter		Typ	Beschreibung	
bufnum		UINT	Nummer des Puffers, der freigegeben werden soll (0: alle freigeben)	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	0 Funktion erfolgreich 1 Ungültiges Handle oder Funktion aus einem anderen Grund fehlgeschlagen	

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCFreeBuffer
VAR_INPUT
  bufnum    : UINT;
END_VAR
VAR_OUTPUT
  retval    : DINT;
END_VAR;
```

1.7.3.15 VNCFreeOldSockets

Diese Funktion steht nur beim Betriebssystem RTK zur Verfügung.

Sockets die nicht schnell genug freigegeben werden, werden mit xn_abort geschlossen.

Übergabeparameter		Typ	Beschreibung	
enable		UINT	Freigeben aktivieren oder deaktivieren	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	0 Funktion erfolgreich 1 Ungültiges Handle oder Funktion aus einem anderen Grund fehlgeschlagen	

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCFreeOldSockets
VAR_INPUT
  enable    : UINT;
END_VAR
```

```
VAR_OUTPUT
  retval    : DINT;
END_VAR;
```

1.7.3.16 VNCGetSize

Diese Funktion ruft die Höhe und Breite des Remote-Desktops ab.

Übergabeparameter		Typ	Beschreibung	
hVNC		_VNCHANDLE	Von vncconnect zurückgegebenes Verbindungs-Handle	
width		^DINT	Zeiger auf eine DINT	
height		^DINT	Zeiger auf eine DINT	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	0	Funktion erfolgreich
			#0	Ungültiges Handle angegeben oder Funktion aufgrund eines anderen Fehlers fehlgeschlagen

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCGetSize
VAR_INPUT
  hVNC      : _VNCHANDLE;
  width     : ^DINT;
  height    : ^DINT;
END_VAR
VAR_OUTPUT
  retval    : DINT;
END_VAR;
```

1.7.3.17 VNCReset

Diese Funktion muss beim Zurücksetzen der SPS aufgerufen werden. Der Zeiger auf die Callback-Funktion wird gelöscht. Dies ist notwendig, da die Callback-Funktion nur gültig ist, wenn die CPU in RUN_RAM oder RUN_ROM Status ist. Die Funktion trennt allen Clients durch den Aufruf von [VNCExit\(\)](#).

(In der Regel muss diese Funktion nicht aus einer LASAL-Funktion aufgerufen werden, das OS macht dies beim Zurücksetzen automatisch.)

1.7.3.18 VNCSendHIDEEvent

Diese Funktion sendet Touch/Maus-Events an den VNC-Server. Die MyIO::GetEvent Funktion ist der ideale Ort im LASAL Source-Code, um diese Funktion aufzurufen.

Übergabeparameter		Typ	Beschreibung
hVNC		_VNCHANDLE	Von vncconnect zurückgegebenes Verbindungs-Handle
eventtype		UDINT	_EVENT_HIDMOVE _EVENT_HIDPRESS or _EVENT_HIDRELEASE
button		UINT	Tastencode (gleich wie in der _EVENT Struktur)
x		INT	X-Position, wo das Event stattgefunden hat
y		INT	Y-Position, wo das Event stattgefunden hat
Rückgabeparameter		Typ	Beschreibung
retval		DINT	0 Funktion erfolgreich #0 Ungültiger Handle angegeben oder Funktion fehlgeschlagen

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCSendHIDEEvent
VAR_INPUT
  hVNC      : _VNCHANDLE;
  eventtype : UDINT;
  button    : UINT;
  x         : INT;
  y         : INT;
END_VAR
VAR_OUTPUT
  retval    : DINT;
END_VAR;
```

1.7.3.19 VNCSendKbrdEvent

Diese Funktion sendet Tastatur-Events an dem VNC-Server. Die MyIO::GetEvent() Funktion ist eine ideale Lage in dem LASAL Source-Code, um diese Funktion aufzurufen.



Wenn es Tasten mit den OS-Funktionen gibt, wie z.B. Pfeiltasten zur Änderung des Focus, ist es praktisch zusätzliche Tasten oder ein Menü zur Verfügung zu stellen, die die Verwendung dieser Tasten auch ermöglicht.

Übergabeparameter	Typ	Beschreibung	
_hVNC	_VNCHANDLE	Von vncconnect zurückgegebenes Verbindungs-Handle	
eventtype	UDINT	_EVENT_KEYPRESS or _EVENT_KEYRELEASE	
scancode	UINT	Scan-Code (wie in der Struktur _EVENT)	
modifier	UINT	Modifikator (wie in der _EVENT Struktur)	
Übergabeparameter	Typ	Beschreibung	
retval	DINT	0	Funktion erfolgreich
		#0	Ungültiger Handle angegeben oder Funktion fehlgeschlagen

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCSendKbrdEvent
VAR_INPUT
    _hVNC      : _VNCHANDLE;
    eventtype  : UDINT;
    scancode   : UINT;
    modifier   : UINT;
END_VAR
VAR_OUTPUT
    retval     : DINT;
END_VAR;
```

1.7.3.20 VNCSendKeyboardEvents

Diese Funktion wird verwendet, um festzulegen oder abzufragen, ob der VNC-Client Tastatur-Events an den VNC-Server senden soll.

Übergabeparameter	Typ	Beschreibung	
-------------------	-----	--------------	--

<u>_hVNC</u>	<u>_VNCHANDLE</u>	Von vncconnect zurückgegebenes Verbindungs-Handle
set	DINT	0 Wert erhalten 1 Wert einstellen
pSend	^DINT	Zeiger auf einen Integer, wo der Wert gespeichert wird
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich #0 Ungültiges Handle oder Funktion aufgrund eines anderen Fehlers fehlgeschlagen

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSendKeyboardEvents
VAR_INPUT
  _hVNC      : _VNCHANDLE;
  set        : DINT;
  pSend      : ^DINT;
END_VAR
VAR_OUTPUT
  retval     : DINT;
END_VAR;
```

1.7.3.21 VNCSendPointerEvents

Diese Funktion wird verwendet, um festzulegen oder abzufragen, ob der VNC-Client Zeiger-Events (Maus oder TouchScreen) an den VNC-Server senden soll.

Übergabeparameter	Typ	Beschreibung
<u>_hVNC</u>	<u>_VNCHANDLE</u>	Von vncconnect zurückgegebenes Verbindungs-Handle
set	DINT	0 Wert erhalten 1 Wert einstellen
pSend	^DINT	Zeiger auf einen Integer, wo der Wert gespeichert wird
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich

		#0	Ungültiges Handle oder Funktion aufgrund eines anderen Fehlers fehlgeschlagen
--	--	----	-------------------------------------------------------------------------------

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSendPointerEvents
VAR_INPUT
    _hVNC      : _VNCHANDLE;
    set        : DINT;
    pSend      : ^DINT;
END_VAR
VAR_OUTPUT
    retval     : DINT;
END_VAR;
```

1.7.3.22 VNCSetCliSleep

Diese Funktion lässt den VNC-Client-Thread schlafen, um die CPU-Last zu senken.

Übergabeparameter	Typ	Beschreibung	
_hVNC	_VNCHANDLE	Von vncconnect zurückgegebenes Verbindungs-Handle	
sstate	UINT	0	Nachrichten verarbeiten fortsetzen
		1	Thread schlafen schicken
kalive	UINT	0 Kein Event wird generiert 1 Im sleep-Modus wird alle 10 s ein Event an den VNC-Server gesendet um ein Timeout (Default 3600 s) zu verhindern	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	Betriebssystem RTK sstate Funktion erfolgreich #0 Ungültiges Handle oder Funktion aufgrund eines anderen Fehlers fehlgeschlagen	
		Andere Betriebssysteme	
		0	Funktion erfolgreich

		#0 Ungültiges Handle oder Funktion aufgrund eines anderen Fehlers fehlgeschlagen
--	--	-------------------------------------------------------------------------------------

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCSetCliSleep
VAR_INPUT
    _hVNC    : _VNCHANDLE;
    sstate   : UINT;
    kalive   : UINT;
END_VAR
VAR_OUTPUT
    retval   : DINT;
END_VAR;
```

1.7.3.23 VNCSetConnectTimeout

Diese Funktion gibt den Timeout für den Verbindungsaufbau an. Sie steht beim Betriebssystem RTK ab der Version 1.1.27 der VNC-Client-DLL zur Verfügung.

Übergabeparameter	Typ	Beschreibung
Timeout	UDINT	Zeit in Sekunden
Rückgabeparameter	Typ	Beschreibung
retval	DINT	Immer 0

Prototyp

```
FUNCTION __CDECL GLOBAL G_ VNCSetConnectTimeout
VAR_INPUT
    Timeout   : UDINT;
END_VAR
VAR_OUTPUT
    retval   : DINT;
END_VAR;
```

1.7.3.24 VNCSetKaValues

Mit dieser Funktion wird das Versenden von Keep-Alive-Paketen konfiguriert. Keep-Alive-Pakete werden nach einem Kommunikationsfehler vom VNC-Client an den VNC-Server gesendet. Wenn auf einer Verbindung keine Pakete mehr ausgetauscht werden, dann werden Keep-Alive-Pakete gesendet, um zu prüfen, ob der Kommunikationspartner noch verbunden ist.

Übergabeparameter	Typ	Beschreibung	
_hVNC	_VNCHANDLE	Von vncconnect zurückgegebenes Verbindungs-Handle	
usInterval	UINT	Wartezeit in Sekunden, bis das erste Keep-Alive-Paket nach einem Kommunikationsfehler gesendet wird. Geben Sie 0 an, um keine Keep-Alive-Pakete zu senden.	
usRetry	UINT	Zeit in Sekunden zwischen den Keep-Alive-Paketen	
usTimeout	UINT	Innerhalb dieser Zeit in Sekunden müssen Keep-Alive-Pakete beantwortet sein. Danach wird ein Fehler gesetzt und die Verbindung unterbrochen.	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	Funktion erfolgreich
		#0	Ungültiges Handle oder Funktion aufgrund eines anderen Fehlers fehlgeschlagen

Prototyp

```
FUNCTION __cdecl GLOBAL G_ VNCSetKaValues
VAR_INPUT
    _hVNC      : _VNCHANDLE;
    usInterval : UINT;
    usRetry    : UINT;
    usTimeout  : UINT;
END_VAR
VAR_OUTPUT
    retval     : DINT;
END_VAR;
```

1.7.3.25 VNCSetOptions

Diese Funktion steht beim Betriebssystem RTK nicht zur Verfügung.

Mit dieser Funktion kann der VNC-Client konfiguriert werden.



Die Werte für VNCCLI_OPTION_ENCODING_FORMAT werden erst beim Verbindungsaufbau übernommen.

Übergabeparameter	Typ	Beschreibung
-------------------	-----	--------------

optionId	DINT	ID des Parameters, der gesetzt werden soll (siehe Tabelle)
optionAddr	^void	Adresse, wo der Parameterwert hinterlegt ist
optionSize	DINT	Größe des Parameterwerts in Byte (siehe Tabelle)
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich -1 Ungültige optionAddr oder ungültige optionSize oder optionId unbekannt

Prototyp

```
FUNCTION __CDECL GLOBAL G_ VNCSetOptions
VAR_INPUT
  optionId      : DINT;
  optionAddr    : ^void;
  optionSize    : DINT;
END_VAR
VAR_OUTPUT
  retval       : DINT;
END_VAR;
```

Mögliche Optionen

optionId	optionSize	Beschreibung
VNCCLI_OPTION_USE_REMOTE_CURSOR (1)	4	0 Remote-Cursor wird nicht lokal angezeigt 1 Remote-Cursor wird lokal angezeigt
VNCCLI_OPTION_COLOR_DEPTH (2)	4	Zulässige Werte sind nur 16 und 24.
VNCCLI_OPTION_ENCODING_FORMAT (3)	Länge des ASCII-Strings plus 1	Die Optionswerte sind als ASCII-Strings anzugeben. • Tight • ZRLE • Ultra • Hextile • Zlib • CoRRE • RRE • Raw

VNCCLI_OPTION_CLEAR_SCREEN_AT_CONNECT (4)	4	<p>Wenn inaktiv (vor dem Verbinden mit dem VNC-Server auf 0 gesetzt), wird der Bildschirm beim Verbindungsaufbau nicht gelöscht. Es wird das letzte sichtbare Bild angezeigt, bis die erste Aktualisierung vom VNC-Server empfangen wird.</p> <p>Wenn aktiv (vor dem Verbinden mit dem VNC-Server auf 1 gesetzt), wird der Bildschirm beim Verbindungsaufbau gelöscht. Es wird ein schwarzer Bildschirm angezeigt, bis die erste Aktualisierung vom VNC-Server empfangen wird.</p>												
VNCCLI_OPTION_SCALING_FACTOR (5)	4	<p>Skalierungsfaktor für den VNC-Server.</p> <table border="1" style="margin-left: 20px;"> <tr><td>0</td><td>Keine Skalierung</td></tr> <tr><td>1</td><td>1:1 (100 %)</td></tr> <tr><td>2</td><td>1:2 (50 %)</td></tr> <tr><td>3</td><td>1:3 (33 %)</td></tr> <tr><td>4</td><td>1:4 (25 %)</td></tr> <tr><td colspan="2">usw.</td></tr> </table> <p>Wichtige Hinweise:</p> <p>Der Skalierungsfaktor muss vor dem Verbindungsaufbau gesetzt werden.</p> <p>Der Skalierungsfaktor wird erst ab den OS-Versionen Salamander 09.02.163 bzw. 09.04.023 unterstützt.</p> <p>Der Skalierungsfaktor wird nicht von allen VNC-Servern unterstützt. VNC-Server von SIGMATEK unterstützen den Skalierungsfaktor.</p>	0	Keine Skalierung	1	1:1 (100 %)	2	1:2 (50 %)	3	1:3 (33 %)	4	1:4 (25 %)	usw.	
0	Keine Skalierung													
1	1:1 (100 %)													
2	1:2 (50 %)													
3	1:3 (33 %)													
4	1:4 (25 %)													
usw.														
VNCCLI_OPTION_USE_COLOR_DEPTH_FROM_SERVER (6)	4	<p>Wenn diese Option vor dem Verbindungsaufbau gesetzt wird, dann wird die Standardfarbtiefe des Servers verwendet.</p>												

1.7.3.26 VNCSetTaskDelayTime

Mit dieser Funktion wird die Zeit eingestellt, um die der VNC-Client-Thread in jedem Durchlauf blockiert wird.

Übergabeparameter	Typ	Beschreibung	
_hVNC	_VNCHANDLE	Von vncconnect zurückgegebenes Verbindungs-Handle	
delayTime	UINT	Zeit, um die der VNC-Client-Thread in jedem Durchlauf blockiert wird, in ms	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0 Funktion erfolgreich	

		#0 Ungültiges Handle oder Funktion aufgrund eines anderen Fehlers fehlgeschlagen
--	--	-------------------------------------------------------------------------------------

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCSetTaskDelayTime
VAR_INPUT
    _hVNC      : _VNCHANDLE;
    delayTime  : UINT;
END_VAR
VAR_OUTPUT
    retval     : DINT;
END_VAR;
```

1.7.3.27 Fehlermeldungen in der Logdatei Event00.log

Der VNC Client schreibt je nach Debug Level verschiedene Fehlermeldungen in die Datei Event00.log.

Fehlermeldung	Log-Level	Beschreibung
logParams: %s	Info	Initialisierung der Logging-Parameter
VNC Client initialized	Info	Der VNC Client wurde initialisiert
VNC client tries to set up a connection to host: %s (LASAL-Viewer: 0x%p)	Info	Der VNC-Client startet nun die Verbindung
VNC client - disconnect <%s>	Info	Ein VNC Client wurde getrennt
VNC-Client service started: <%s> : <%s> : 0x%p	Error	Mehr Informationen über den Status des VNC Clients
VNC Client initialization failed: not enough memory!	Error	Es steht nicht genug OSHEAP für den VNC Client zur Verfügung
VNCConnect failed : Couldn't create VNC-Viewer-thread!!	Error	Der VNC Viewer Thread konnte nicht gestartet werden
VNCConnect failed : VNC Client not initialized!	Error	Der VNC Client wurde nicht ordnungsgemäß initialisiert
VNCDisconnect for host <%s> failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
VNCGetSize for handle 0x%08x failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
VNCSendKeyboardEvents for handle 0x%08x failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden

VNCSendPointerEvents for handle 0x%08x failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
VNCSendHIDEEvent for handle 0x%08x failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
VNCSetCliSleep for handle 0x%08x failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
VNCFreeBuffer failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
VNCSetTaskDelayTime for handle 0x%08x failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
VNCFreeOldSockets failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
VNCDraw for handle 0x%08x failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
VNCForceRepaint for handle 0x%08x failed: No pViewManager!	Error	Der interne pViewManager Thread konnte nicht gestartet werden
EXCEPTION WHILE COPYING RECTS	Error	Es ist ein interner Fehler aufgetreten beim Kopieren der Bilddaten
DIBSectionBuffer::recreateBuffer failed: NOT ENOUGH MEMORY for Pixelbuffer (%i Bytes)	Error	Es steht nicht genug Speicher zur Verfügung um die Bilddaten zwischenzuspeichern
refresh palette called for truecolour DIB	Info	Die Farbpalette wurde neu geladen
unable to connect to %s (%s)	Info	Es konnte keine Verbindung aufgebaut werden
Connected to VNC-Server on host %s.	Info	Die Verbindung wurde erfolgreich aufgebaut
Authentication failure : %s : reason: %s	Info	Es ist ein Fehler bei der Authentifizierung aufgetreten
Disconnecting host <%s>, Handle: %p	Info	Die Verbindung wurde beendet
No such host (%s)	Info	Der Host wurde nicht gefunden
stop all: No connections	Info	Zur Zeit existieren keine Verbindungen
stop all: disconnected %i hosts	Info	Anzahl der getrennten Hosts
SetCliSleep state:%d, keepalive:%d	Info	Die Option SetCliSleep wurde geändert
SetTaskDelayTime %d	Info	Die Option SetTaskDelayTime wurde geändert
FreeOldSockets %s	Info	VNC Sockets wurden aus der CLOSED_WAIT Liste gelöscht

CViewThread::VNCGetSize: VNC_NO_ATTACHED_VIEW	Error	Es ist kein View mit dem VNC-Client verbunden
CViewThread::VNCGetSize: VNC_NO_ATTACHED_VIEW	Error	Es ist kein View mit dem VNC-Client verbunden
CViewThread::VNCSendHIDEEvent: VNC_NO_ATTACHED_VIEW	Error	Es ist kein View mit dem VNC-Client verbunden
CViewThread::VNCDraw: VNC_NO_ATTACHED_VIEW	Error	Es ist kein View mit dem VNC-Client verbunden
CViewThread::VNCForceRepaint: VNC_NO_ATTACHED_VIEW	Error	Es ist kein View mit dem VNC-Client verbunden
CViewThread::VNCSendKeyboardEvents: VNC_NO_ATTACHED_VIEW	Error	Es ist kein View mit dem VNC-Client verbunden
CViewThread::VNCSendPointerEvents: VNC_NO_ATTACHED_VIEW	Error	Es ist kein View mit dem VNC-Client verbunden
RFB Exception in CViewThread::run(): %s	Error	Es ist ein interner Fehler aufgetreten
CViewThread::run() : inner rdr::Exception: %s	Error	Es ist ein interner Fehler aufgetreten
Failed to create CViewThread for host <%s>	Error	CViewThread konnte nicht erstellt werden
closing - %s	Info	Verbindung schließen
CView::CView() : FAILED TO CREATE DIBSectionBuffer (buffer for remote-picture data)	Error	Der Bildspeicher konnte nicht angelegt werden
%s: LockBuffer FAILED -> Closing	Error	Der Lock für den Bildspeicher konnte nicht gesetzt werden
getUserPasswd failed: missing username or password.	Error	Der Benutzername oder das Passwort stimmen nicht überein

1.7.3.28 Fehlermeldungen VNC Client

Funktionen mit einem DINT als Rückgabeparameter sind fehlgeschlagen, wenn sie einen Wert größer als 0 zurückliefern.

Symbolname	Wert	Beschreibung
VNC_OK	0	Funktion war erfolgreich. Keine Fehler.
VNC_NO_VIEWMAN GER	1	Der Main-Thread des VNC-Client wurde unerwartet beendet.
VNC_NO_THREAD	2	Der Thread, der diese Verbindung bearbeitet, wurde unerwartet beendet.

VNC_NO_BUFFER	3	Der Remote-Pixel Puffer ist noch nicht verfügbar. Funktion VNCGetSize
VNC_NO_ATTACHED_VIEW	4	Das Objekt dieser Verbindung wurde unerwartet beendet.
VNC_FAILED_BUFFER_LOCK	5	Derzeit nicht verwendet
VNC_FAILED_BUFFER_UNLOCK	6	Ein Fehler ist bei der Signalisierung des Remote-Pixel Puffer Flags aufgetreten.
VNC_INIT_FAILED	7	Nicht genügend Speicher zum Initialisieren des VNC-Client. Funktion: VNCInit()
VNC_NOT_DRAWN	8	Dieser Fehler kann auftreten, wenn der VNC-Client nicht über genügend Speicher zum Zuweisen des Pixel-Puffers verfügt, um das Remote-Bild zu erhalten.

1.7.4 Command Line Interface (CLI) Befehle

Der VNC-Client-Dienst kann durch den VNCCLI-Befehl manuell gestartet oder gestoppt werden:

Syntax

`vnccli start | stop | version`

Beispiel

`vnccli start`

Startet den VNC-Client Service.

`vnccli stop`

Stoppt den VNC-Client Service.

`vnccli version`

Zeigt die DLL-Version und der VNC-Protokoll-Versionsnummern.

1.8 VNCsvr

1.8.1 Was ist LasalOS VNCsvr?

LasalOS VNCsvr ist eine VNC-Server-Anwendung für das LasalOS. Das VNC-Protokoll ist ein einfaches Protokoll für den Fernzugang von grafischen Anwenderschnittstellen und basiert auf dem Remote-Frame-Puffer (RFB) Konzept. Das Protokoll ermöglicht es einen Server dem Frame-Puffer einfach zu aktualisieren, der über einen Viewer angezeigt wird.

Da er auf der Frame-Puffer-Ebene arbeitet, kann es mit allen Betriebssystemen oder Windows-Anwendungen verwendet werden.

Der VNC-Server ermöglicht Ihnen den Fernzugriff auf Ihren C-IPC von jedem System aus, das die Verwendung eines Standard-VNC-Clients unterstützt (LasalOS, Windows, MacOS, Linux, Unix, ...).

VNC-Clients für Windows, Linux, Unix und andere Systeme sind kostenlos auf den folgenden Webseiten verfügbar: <http://www.realvnc.com> und <http://www.tightvnc.com>.

1.8.2 Der Zweck dieses Dokuments

Dieses Dokument beschreibt die LasalOS-Schnittstelle für den VNC-Server.

Anforderungen

LasalOS	Benötigt LasalOS 1.1.3
Platform	LasalOS VNCsvr ist verfügbar auf dem C-IPC, ETV und EDGE.

Einbauzustand

- Um den VNC-Server mit einem C-IPC zu verwenden, kopieren Sie die Dateien VNCSV.R.DLM und ZLIB.DLM in das Verzeichnis C:\LSLSYS des Zielsystems.
- Kopieren Sie die Datei VNCPWD.TXT in das Verzeichnis C:\ auf das Zielsystem. Das in der Datei gespeicherte Kennwort ist verschlüsselt.
- Bearbeiten Sie die AUTOEXEC.LSL und fügen Sie die Zeile 'VNCSV START' ein, um den VNC-Server-Dienst zu starten, wenn der C-IPC hochfährt.



Terminals mit Edge PLC sind mit einer niedrigen OS-Heap-Einstellung (Speicher, reserviert für das Betriebssystem) konfiguriert. Wenn VNC verwendet wird, wird empfohlen, den Heap des Betriebssystems zu vergrößern. (Z.B. 10000 Byte.)

Beispiel: `SET OSHEAP 10000`

-
- Das Standard-Passwort Ihrer VNC-Server ist "SIGMATEK" - um dieses Passwort zu ändern, verwenden Sie die folgenden CLI-Befehl VNCSV SETPASS sigmatek <your new password>
Der VNC-Server muss mit dem Befehl VNCSV START initialisiert worden sein, bevor Sie ein neues Passwort setzen. (Dies kann entweder über die [AUTOEXEC.LSL](#) oder durch Eingabe des Befehls in der CLI erfolgen).

- Nach dem Aufruf von 'VNCSV START' ist der VNC-Server einsatzbereit und kann in der CLI konfiguriert werden. Die Funktionen der OS-Oberfläche sind nur notwendig, wenn Sie eine benutzerfreundliche Oberfläche bereitstellen möchten.

Der VNC-Server kann durch einfache Eingabe von 'VNCSV STOP' in der CLI gestoppt werden.



- Die Repeater-Funktion wird über die Datei vncsvr.cfg konfiguriert. Wenn diese Datei verfügbar ist (im Ordner C:\LslSys\ auf der Steuerung) und alle benötigten Einträge (siehe [Konfiguration des Repeaters](#)) gültig sind, wird die Repeater-Funktion automatisch mit dem VNC-Server gestartet.

1.8.3 LASAL OS VNCSV-API Funktionen

Die Betriebssystemschnittstelle "VNCSV" ist verfügbar, wenn der VNC-Server die Benutzerplattform unterstützt und der Dienst durch Hinzufügen von VNCSV START zur AUTOEXEC.LSL gestartet wird. Die Schnittstellestruktur, die Funktionsprototypen und die Konstanten sind in der lsl_st_vnc.h Datei verwendet.

1.8.3.1 Die OS-Schnittstelle VNCSV

```

TYP
OS_VNCSV
  udSize : STRUCT
  udVersion : UDINT;
  udVersionVNC : UDINT;
  VNCSVRIInit : pvoid;
  VNCSVRExit : pvoid;
  VNCSVRReset : pvoid;
  VNCSVRSetPass : PVOID;
  VNCSVREnumConnections : PVOID;
  VNCSVRDisconnectAllClients : PVOID;
  VNCSVRDisconnectClient : PVOID;
  VNCSVRFullRefreshCycle : PVOID;
  VNCSVRTimeoutInactiveClient : PVOID;
  VNCSVRAcceptKeyboardEvents : PVOID;
  VNCSVRAcceptPointerEvents : PVOID;
  VNCSVRSetFilter : PVOID;
  VNCSVRDisconnectClientsOnNonsharedConnection : pvoid;
  VNCSVRNeverShared : pvoid;
  VNCSVRAAlwaysShared : pvoid;
  VNCSVRMaxConnections : pvoid;
  VNCSVRCClientWaitTimeMillis : pvoid;
  VNCSVRCompareFB : pvoid;
  VNCSVRSetDisplayOffset : pvoid;

```

```

VNCSVRUpdateRect           : pvoid;
VNCSVRIinitialized         : pvoid;
VNCSVRConnectedClients     : pvoid;
VNCSVRReserved0            : pvoid;
VNCSVRReserved1            : pvoid;
VNCSVRDisableSharedMemory  : pvoid;
VNCSVREnumConnections2     : pvoid;
VNCSVRConnectedClients2    : pvoid;
END_STRUCT;
END_TYPE

```

Im Folgenden finden Sie ein kurzes Beispiel für die Verwendung dieser Schnittstelle.

Beispiel

```

...
VAR
  pVNCSvr    : ^OS_VNCSVR;
  retv       : SYS_ERROR;
END_VAR;

  retv := OS_CILGet( "VNCSVR" , #pVNCSvr$void);

  IF pVNCSvr = NIL THEN
    TRACE( "No VNCSVR-Interface! Check your OS-Version-Number! " );
  ELSE
    err := pVNCSvr^.VNCSVRExit $ G_VNCSVRExit ();
  END_IF;

  OS_CILRelease( "VNCSVR" );
...

```

1.8.3.2 udSize

Dies ist eine einfache UDINT Variable, die der Größe der Struktur-Schnittstelle enthält.

1.8.3.3 udVersion

Dies ist eine einfache UDINT-Variable, die die DLL-Version der VNC-Server-DLL enthält.

Derzeit wird nur das untere Word verwendet, um die dreiteilige DLL-Versionsnummer zu definieren (#1.#2.#3):

HIGH WORD				LOW WORD			
31-28	27-24	23-20	19-13	15-12	11-8	7-4	3-0
Nicht verwendet (0)				#1	#2	#3	

Beispiel

Die DLL-Version 1.1.1 entspricht dem Hex-Wert 16#00001101.

1.8.3.4 udVersionVNC

Dies ist eine einfache UDINT Variable, die die Versionsnummer des VNC-Protokolls enthält.

Das obere Wort enthält die Hauptversionsnummer, das untere Wort enthält die niedrige Versionsnummer.

Beispiel

VNC-Protokoll Version 3.8 entspricht den Hex-Wert 16#00030008.

1.8.3.5 VNCSVRAcceptKeyboardEvents

Mit dieser Funktion kann angegeben werden, ob der VNC-Server einer Remote-Tastatur Zugriff ermöglicht. Der Default-Wert ist TRUE.

Übergabeparameter	Typ	Beschreibung		
set	DINT	Parameter wird verwendet, um einen Wert zu setzen oder abzurufen <table border="1" data-bbox="492 872 1041 976"> <tr> <td>0 Wert abrufen</td></tr> <tr> <td>1 Wert setzen</td></tr> </table>	0 Wert abrufen	1 Wert setzen
0 Wert abrufen				
1 Wert setzen				
pValue	^DINT	Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll		
Übergabeparameter	Typ	Beschreibung		
retval	DINT	0 Funktion erfolgreich		

Siehe auch: [VNC SVR ACCEPT KEYBOARD](#)

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVRAcceptKeyboardEvents
VAR_INPUT
  set      : DINT;
  pAccept : ^DINT;
```

```

END_VAR
VAR_OUTPUT
    retval
        : DINT;
END_VAR;

```

1.8.3.6 VNC_SVRAcceptPointerEvents

Mit dieser Funktion kann festgelegt werden, ob der VNC-Server den Fernzugriff auf Geräte (Touchscreen & Maus) erlaubt. Der Default-Wert ist TRUE.

Übergabeparameter	Typ	Beschreibung
set	DINT	Parameter wird verwendet, um einen Wert zu setzen oder abzurufen <div style="display: flex; justify-content: space-around; align-items: center;"> 0 Wert abrufen 1 Wert setzen </div>
pAccept	^DINT	Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll
Übergabeparameter	Typ	Beschreibung
retval	DINT	<div style="display: flex; justify-content: space-around; align-items: center;"> 0 Funktion erfolgreich </div>

Siehe auch: [VNC_SVR_ACCEPT_POINTER](#)

Prototyp

```

FUNCTION __cdecl GLOBAL G_VNC_SVRAcceptPointerEvents
VAR_INPUT
    set          : DINT;
    pAccept     : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;

```

1.8.3.7 VNC_SVRAlwaysShared

Diese Funktion legt fest, ob der Server geteilte Verbindungen immer erlaubt. Wenn auf 1 gesetzt, lässt der Server immer gemeinsame Verbindungen zu, unabhängig davon, was der Client definiert hat. Die Default-Einstellung ist 0.

Übergabeparameter	Typ	Beschreibung

set	DINT	Parameter wird verwendet, um einen Wert zu setzen oder abzurufen
		0 Wert abrufen
		1 Wert setzen
pValue	^DINT	Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich

Siehe auch: [VNC_SVR_ALWAYS_SHARED](#)

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVRAAlwaysShared
VAR_INPUT
    set          : DINT;
    pValue       : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.8 VNCVRClientWaitTimeMillis

Diese Eigenschaft legt fest, wie lange (in Millisekunden) der Server auf einen Client wartet, der nicht antwortet; der Standardwert ist 20000.

Übergabeparameter	Typ	Beschreibung
set	DINT	Parameter wird verwendet, um einen Wert zu setzen oder abzurufen
		0 Wert abrufen
		1 Wert setzen
pValue	^DINT	Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich

Siehe auch: [VNC_SVR_WAIT_FOR_CLIENT](#)

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCVRClientWaitTimeMillis
VAR_INPUT
    set          : DINT;
    pValue       : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.9 VNCVRCompareFB

Diese Eigenschaft legt fest, ob der Server einen Pixelvergleich auf dem Bildpuffer durchführt, um unnötige Aktualisierungen zu reduzieren (1) oder nicht (0). Der Default-Wert ist 0.

Übergabeparameter	Typ	Beschreibung
set	DINT	Parameter wird verwendet, um einen Wert zu setzen oder abzurufen <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> 0 Wert abrufen 1 Wert setzen </div>
pValue	^DINT	Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll
Rückgabeparameter		
retval	DINT	0 Funktion erfolgreich

Siehe auch: [VNC_SVR_COMPARE_FB](#)

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCVRCompareFB
VAR_INPUT
    set          : DINT;
    pValue       : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.10 VNCSVRDisconnectAllClients

Diese Funktion trennt alle VNC-Clients vom Server, aber der Server bleibt aktiv.

Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVRDisconnectAllClients
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.11 VNCSVRDisconnectClient

Diese Funktion trennt einen angegebenen Client

Übergabeparameter	Typ	Beschreibung
strClient	^CHAR	Dieser Parameter enthält den zu trennenden Client (z.B.: 192.168.10.14::3601). Es kann einer der in der Aufzählungs-Callback-Funktion zurückgegebenen Strings verwendet werden.
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVRDisconnectClient
VAR_INPUT
    strClient      : ^CHAR
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.12 VNCSVRConnectedClients

Diese Funktion gibt die Anzahl der mit dem VNC-Server verbundenen Clients zurück

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVRConnectedClients
```

```
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

1.8.3.13 [VNCSVRCConnectedClients2](#)

Diese Funktion ist die gleiche wie die Funktion [VNCSVRCConnectedClients\(\)](#) mit dem zusätzlichen Parameter `listenerPort`. Dieser Parameter erlaubt es, nur Verbindungen zu berücksichtigen, die über den angegebenen Port des Servers aufgebaut wurden (die Port-Nummer des Servers wird in der Umgebungsvariablen [VNC_SVR_PORT](#) bzw. [VNC_SVR_PORT2](#) angegeben).

Übergabeparameter	Typ	Beschreibung
listenerPort	DINT	Legt die Port-Nummer des Servers fest. Wenn dieser Parameter > 0 ist, werden nur die über diesen Port aufgebauten Verbindungen berücksichtigt. Der Wert 0 bedeutet, dass alle Verbindungen berücksichtigt werden.
Rückgabeparameter	Typ	Beschreibung
retval	DINT	

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCSVRCConnectedClients2
VAR_INPUT
  listenerPort      : DINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

1.8.3.14 [VNCSVRDisconnectClientsOnNonsharedConnection](#)

Verwenden Sie diese Funktion, um die Eigenschaft `DisconnectClientsOnNonsharedConnection` zu setzen oder abzurufen. Wenn diese Eigenschaft auf 1 gesetzt ist, trennt der Server die Verbindung zu allen anderen Clients, wenn sich ein Client verbindet, bei dem die Option "Gemeinsame Verbindung" nicht gesetzt ist. Wenn die Eigenschaft auf 0 gesetzt ist, dürfen sich nur Clients verbinden, bei denen die Option "Gemeinsame Verbindung" gesetzt ist, und Clients mit der Option "nicht gemeinsame Verbindung" werden abgewiesen.

Übergabeparameter	Typ	Beschreibung

set	DINT	Parameter wird verwendet, um einen Wert zu setzen oder abzurufen
		0 Wert abrufen 1 Wert setzen
pDisconnect	^DINT	Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich

Siehe auch: [VNC_SVR_DISCONNECT_ON_NONSHARED](#)

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCVRDisconnectClientsOnNonsharedConnection
VAR_INPUT
    set          : DINT;
    pDisconnect : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.15 VNCVRDisableSharedMemory

Diese Funktion kann die Vollbildaktualisierung aus dem gemeinsamen Speicher verhindern, wenn sich mehr als 75 % des Bildschirminhalts geändert haben.

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCVRDisableSharedMemory
VAR_INPUT
    set          : DINT;
    pValue      : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Diese Funktion wird ab der Schnittstellenversion 01.01.037 unterstützt.



Diese Funktionalität wird nur unter Salamander unterstützt.

1.8.3.16 VNCSVREnumConnections

Die bereitgestellte Callback-Funktion wird für jeden verbundenen VNC-Client einmal aufgerufen.

Der Parameter pThis wird bereitgestellt, damit eine Member-Funktion von der globalen Callback-Funktion aufgerufen werden kann. Dieser kann als Zeiger auf eine Klasse verwendet werden, die die Member-Funktion aufrufen kann. Nach der letzten Verbindung wird die Callback-Funktion mit einem NULL-Zeiger als hostinfo-Parameter aufgerufen, um das Ende der Liste anzuzeigen.

Übergabeparameter	Typ	Beschreibung
pThis	PVOID	Zeiger auf das LASAL-Klassenobjekt, das diese Verbindung behandelt
callback	PVOID	Zeiger auf eine globale Aufzählungs-Callback-Funktion
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich

Callback-Prototyp

```
FUNCTION __cdecl GLOBAL VNCSVREnumConnections
VAR_INPUT
    pThis      : pvoid; (* arbitrary Pointer - z.B. eine LASAL-Klasse *)
    hostinfo   : ^CHAR; (* "IPAddr::Port" z.B. 192.168.10.15::7600 *)
END_VAR;
```

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCSVREnumConnections
VAR_INPUT
    pThis      : pVoid
    Callback   : pVoid
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;
```

1.8.3.17 VNCSVREnumConnections2

Diese Funktion ist gleich wie die Funktion [VNCSVREnumConnections\(\)](#) mit dem zusätzlichen Parameter `listenerPort`. Dieser Parameter erlaubt es, nur Verbindungen aufzulisten, die über den angegebenen Port des Servers aufgebaut wurden (die Port-Nummer des Servers wird in der Umgebungsvariablen [VNC_SVR_PORT](#) bzw. [VNC_SVR_PORT2](#) angegeben).

Übergabeparameter	Typ	Beschreibung
<code>pThis</code>	PVOID	Zeiger auf das Objekt der LASAL-Klasse, das diese Verbindung behandelt
<code>Callback</code>	PVOID	Zeiger auf eine globale Aufzählungs-Callback-Funktion
<code>listenerPort</code>	DINT	Legt die Port-Nummer des Servers fest. Wenn dieser Parameter > 0 ist, werden nur die über diesen Port aufgebauten Verbindungen aufgelistet. Der Wert 0 bedeutet, dass alle Verbindungen aufgelistet werden.
Rückgabeparameter	Typ	Beschreibung
<code>retval</code>	DINT	0 Funktion erfolgreich

Callback Prototyp

```
FUNCTION __CDECL GLOBAL VNCSVREnumConnections
VAR_INPUT
    pThis      : PVOID; (* beliebiger Zeiger - z.B. auf eine LASAL-Klasse *)
    hostinfo   : ^CHAR; (* "IPAddr::Port" z.B. 192.168.10.15::7600 *)
END_VAR;
```

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVREnumConnections2
VAR_INPUT
    pThis      : pVoid
    Callback   : pVoid
    listenerPort : DINT;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;
```

1.8.3.18 VNCSVRExit

Diese Funktion trennt die Verbindung aller VNC-Clients und beendet den VNC-Server.

Parameter

Keine

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVRExit;
```

1.8.3.19 VNCSVRFullRefreshCycle

Mit dieser Funktion können die Zeitintervalle festgelegt werden, in denen der VNC-Server ein Vollbild-Update durchführen soll. Der Default-Wert ist 2 Sekunden; 0 deaktiviert diese Funktion.

Übergabeparameter	Typ	Beschreibung		
set	DINT	<p>Parameter wird verwendet, um einen Wert zu setzen oder abzurufen</p> <table border="1" style="margin-left: 20px;"> <tr> <td>0 Wert abrufen</td></tr> <tr> <td>1 Wert setzen</td></tr> </table>	0 Wert abrufen	1 Wert setzen
0 Wert abrufen				
1 Wert setzen				
pSeconds	^DINT	Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll		
		Rückgabeparameter		
retval	DINT	0 Funktion erfolgreich		

Siehe auch: [VNC SVR FULL UPDATE](#)

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVRFullRefreshCycle
VAR_INPUT
    set      : DINT;
    pSeconds : ^DINT;
END_VAR
VAR_OUTPUT
    retval  : DINT;
END_VAR;
```

1.8.3.20 VNCSVRIInit

Diese Funktion initialisiert den VNC-Server. Da diese Funktion automatisch aufgerufen wird, wenn der VNC-Server mit dem CLI-Befehl vncsvr start gestartet wird, wird sie nur dann manuell aufgerufen, wenn der VNC-Server mit der Funktion [VNCSVRExit\(\)](#) beendet wurde.

Übergabeparameter	Typ	Beschreibung
logStr	^CHAR	<p>Dieser Parameter ist ein String, der eine Log-Einstellung definiert. Der Log-String ist in drei Abschnitte unterteilt, die durch Doppelpunkte getrennt sind: „[filter]:[stream]:[log level]“.</p> <p>Der String im ersten Abschnitt definiert einen Modulnamen-Filter ** und protokolliert alle Meldungen von allen Modulen (behalten Sie ** für den normalen Gebrauch.).</p> <p>Der zweite Abschnitt definiert einen Output-Stream; 'file' schreibt die Log-Meldungen in eine Datei (der Dateiname wird mit dem zweiten Parameter dieser Funktion definiert). Andere Stream-Namen könnten aus 'stderr', 'stdout' 'kernel_svr' bestehen.</p> <p>Der dritte Abschnitt definiert das Log-Level. Ein zutreffender logStr würde lauten: <code>":kernel_svr:30"</code>.</p> <div style="text-align: center;">  </div> <p>Verwenden Sie 'kernel_svr' als Log-Stream, um die VNC-Log-Meldungen in die System-Log-Datei c:\sysmsg\event00.log zu schreiben.</p>
LogFileName	^CHAR	<p>Um eine zusätzliche Protokolldatei für den VNC-Server hinzuzufügen, muss "file" als Ausgabestrom im Parameter logStr definiert und ein Name für die Protokolldatei angegeben werden.</p> <p>Beispiel: Für eine Protokolldatei vncsvr.log muss der Server wie folgt initialisiert werden:</p> <p><code>VNCSVRIInit(":file:30", "vncsvr.log")</code></p>

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCSVRIInit
VAR_INPUT
    logStr      : ^CHAR
    logFileName : ^CHAR
END_VAR;
```

Log-Level

Log-Level	Nachrichtentyp	Beschreibung
-----------	----------------	--------------

0	fehler	Schreibt nur Fehler in der Protokolldatei
30	info	Schreibt weitere Informationen zu den Protokollen
100	debug	Nur für das Debugging nötig
150	erweitert	Nur für das Debugging nötig

1.8.3.21 VNCSVRIinitialized

Diese Funktion gibt den Wert 1 zurück, wenn der VNC-Server läuft, sonst 0.

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVRIinitialized
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.22 VNCSVRMaxConnections

Diese Funktion definiert, wie viele Clients gleichzeitig an den Server angeschlossen werden können; der Standardwert ist 4.

Übergabeparameter	Typ	Beschreibung		
set	DINT	Parameter wird verwendet, um einen Wert zu setzen oder abzurufen <table border="1" data-bbox="464 912 1013 999"> <tr> <td>0 Wert abrufen</td></tr> <tr> <td>1 Wert setzen</td></tr> </table>	0 Wert abrufen	1 Wert setzen
0 Wert abrufen				
1 Wert setzen				
pValue	^DINT	Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll		
Rückgabeparameter	Typ	Beschreibung		
retval	DINT	0 Funktion erfolgreich		

Siehe auch: [VNC_SVR_MAX_CONNECTIONS](#)

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCSVRMaxConnections
VAR_INPUT
    set      : DINT;
```

```

pValue      : ^DINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;

```

1.8.3.23 VNCSVRNeverShared

Diese Eigenschaft legt fest, ob der Server gemeinsame Verbindungen zulässt. Wenn auf 1 gesetzt, lässt der Server keine gemeinsamen Verbindungen zu, unabhängig davon, was der Client definiert hat. Die Default-Einstellung ist 0.

Übergabeparameter	Typ	Beschreibung	
set		Parameter wird verwendet, um einen Wert zu setzen oder abzurufen	
		0	Wert abrufen
		1	Wert setzen
pValue		Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll	
Rückgabeparameter	Typ	Beschreibung	
retval		0 Funktion erfolgreich	

Siehe auch: [VNC_SVR_NEVER_SHARED](#)

Prototyp

```

FUNCTION __CDECL GLOBAL G_VNCSVRNeverShared
VAR_INPUT
  set      : DINT;
  pValue   : ^DINT;
END_VAR
VAR_OUTPUT
  retval   : DINT;
END_VAR;

```

1.8.3.24 VNCSVRReset

Nur für den internen Gebrauch. Diese Funktion nicht verwenden!



1.8.3.25 VNCSVRSetDisplayOffset

Nur für den internen Gebrauch. Diese Funktion nicht verwenden!



1.8.3.26 VNCSVRSetFilter

Es kann festgelegt werden, welche Hosts eine Verbindung mit dem VNC-Server herstellen dürfen. Einzelne Regeln können zu einem komplexeren Filter kombiniert werden, indem sie durch Kommas getrennt werden. Es ist wichtig zu beachten, dass die erste passende Regel für einen Host verwendet wird.

Übergabeparameter		Typ	Beschreibung																
strFilter		^CHAR	<p>Dieser Parameter definiert die Filterregeln; die einzelnen Regeln werden durch Komma getrennt.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2">Regel</th><th>Beispiel</th></tr> </thead> <tbody> <tr> <td style="text-align: center;">+<host></td><td>Erlaubt einem angegebenen Host, eine Verbindung herzustellen.</td><td>+192.168.1 0.30</td></tr> <tr> <td style="text-align: center;">+<net>/<net mask></td><td>Erlaubt einem Client aus dem angegebenen Subnetz, sich mit dem Server zu verbinden.</td><td>+192.168.2 0.0 /255.255.25 5.0</td></tr> <tr> <td style="text-align: center;">-<host></td><td>Verbindung mit dem angegebenen Host ist nicht erlaubt.</td><td>+192.168.1 0.30</td></tr> <tr> <td style="text-align: center;">-<net>/<net mask></td><td>Erlaubt Clients aus dem angegebenen Subnetz nicht, sich mit dem Server zu verbinden.</td><td>- 192.168.50. 0 /255.255.25 5.0</td></tr> </tbody> </table>		Regel		Beispiel	+<host>	Erlaubt einem angegebenen Host, eine Verbindung herzustellen.	+192.168.1 0.30	+<net>/<net mask>	Erlaubt einem Client aus dem angegebenen Subnetz, sich mit dem Server zu verbinden.	+192.168.2 0.0 /255.255.25 5.0	-<host>	Verbindung mit dem angegebenen Host ist nicht erlaubt.	+192.168.1 0.30	-<net>/<net mask>	Erlaubt Clients aus dem angegebenen Subnetz nicht, sich mit dem Server zu verbinden.	- 192.168.50. 0 /255.255.25 5.0
Regel		Beispiel																	
+<host>	Erlaubt einem angegebenen Host, eine Verbindung herzustellen.	+192.168.1 0.30																	
+<net>/<net mask>	Erlaubt einem Client aus dem angegebenen Subnetz, sich mit dem Server zu verbinden.	+192.168.2 0.0 /255.255.25 5.0																	
-<host>	Verbindung mit dem angegebenen Host ist nicht erlaubt.	+192.168.1 0.30																	
-<net>/<net mask>	Erlaubt Clients aus dem angegebenen Subnetz nicht, sich mit dem Server zu verbinden.	- 192.168.50. 0 /255.255.25 5.0																	
Rückgabeparameter		Typ	Beschreibung																
RetVal		DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">0</td><td>Funktion erfolgreich</td></tr> <tr> <td style="width: 50%;">#0</td><td>Fehler</td></tr> </table>		0	Funktion erfolgreich	#0	Fehler											
0	Funktion erfolgreich																		
#0	Fehler																		

Siehe auch: [VNC_SVR_FILTER](#)

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCVRSetFilter
VAR_INPUT
    strFilter      : ^CHAR
END_VAR
VAR_OUTPUT
    retval        : DINT;
END_VAR;
```

Beispiel

Der Filterstring -192.168.30.10,+192.168.30.0/255.255.255.0,- lässt jeden Host aus dem Subnetz 192.168.30.xxx zu, außer dem einzelnen Host mit der IP 192.168.30.10 (da die Regel für diesen Host vor der Subnetz-Regel steht). Die letzte Regel (Standardregel) '-' soll jeden anderen Host ausschließen.

Eine weitere Möglichkeit könnte wie folgt aussehen:

Der Filterstring -192.168.30.10,+ lässt alle Hosts außer dem mit der IP 192.168.30.10 zu.



Der Standardfilter ist '+', was bedeutet, dass alle Hosts eine Verbindung zum Server herstellen dürfen!

1.8.3.27 VNCVRSetPass

Diese Funktion wird verwendet um ein neues Passwort für den VNC-Server zu setzen. Um ein neues Passwort zu setzen, muss zunächst das alte zur Authentifizierung eingegeben werden.

Übergabeparameter	Typ	Beschreibung
oldPassword	^CHAR	Zeichenkette, die das alte Passwort enthält
newPassword	^CHAR	Zeichenkette, die das neue Passwort enthält
Rückgabeparameter	Typ	Beschreibung

retval	DINT	VNC_OK OK, das neue Kennwort wurde festgelegt VNC_OLDPWD_IN Authentifizierungsfehler - altes Passwort VALID ist ungültig VNC_INVALID_PW Password-Datei hat ein ungültiges Format DFILE VNC_FAILED_OPE Konnte die Passwort-Datei nicht öffnen N_PWDFILE VNC_FAILED_WRT Konnte nicht in die Passwort-Datei ING_PWDFILE schreiben
--------	------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Prototyp

```
FUNCTION __cdecl GLOBAL G_VNCVRSetPass
    OldPassword      : ^CHAR;
    NewPassword      : ^CHAR;
VAR_OUTPUT
    retval          : DINT;
END_VAR;
```



Das VNC-Server Passwort kann auch eingestellt werden durch Verwenden des folgenden CLI-Befehls:

```
vncsvr setpass <oldpwd> <newpwd>
```

1.8.3.28 VNCSVRTTimeoutInactiveClient

Mit dieser Funktion kann ein Zeitintervall festgelegt werden, in dem der VNC-Server die Verbindung zu einem inaktiven Client trennt. Der Default-Wert ist 3600 Sekunden (der Minimalwert ist 15 Sekunden). Der Default-Wert ist 3600 Sekunden.

Übergabeparameter	Typ	Beschreibung		
set	DINT	Parameter wird verwendet, um einen Wert zu setzen oder abzurufen <table border="1" data-bbox="459 1190 1008 1285"> <tr> <td>0 Wert abrufen</td> </tr> <tr> <td>1 Wert setzen</td> </tr> </table>	0 Wert abrufen	1 Wert setzen
0 Wert abrufen				
1 Wert setzen				
pSeconds	^DINT	Zeiger auf einen Integer, in dem der gewünschte Wert gespeichert werden soll		

Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	Funktion erfolgreich

Siehe auch: [VNC_SVR_TIMEOUT_INACTIVE](#)

Prototyp

```
FUNCTION __CDECL GLOBAL G_VNCVRTimeoutInactiveClient
VAR_INPUT
    set          : DINT;
    pSeconds     : ^DINT;
END_VAR
VAR_OUTPUT
    retval       : DINT;
END_VAR;
```

1.8.3.29 VNCVRUpdateRect

Nur für den internen Gebrauch. Diese Funktion nicht verwenden!



1.8.3.30 Fehler-Codes VNC Server

Funktionen mit einem DINT als Rückgabeparameter sind fehlgeschlagen, wenn sie einen Wert größer als 0 zurückliefern.

Symbolname	Wert	Beschreibung
VNC_OK	16#00	Funktion war erfolgreich. Keine Fehler
VNC_NO SUCH_CLIENT	16#50	Der Client existiert nicht.
VNC_NO_SCREENUPDATER_MANAGER	16#51	Der VNC-Server läuft nicht.
VNC_INVALID_PWDFILE	16#52	Passwort-Datei ist ungültig.
VNC_OLEPWD_INVALID	16#53	Altes Passwort ist ungültig.

VNC_FAILED_OPEN _PWDFILE	16#54	Konnte die Passwort-Datei nicht zum Lesen öffnen.
VNC_FAILED_WRITI NG_PWDFILE	16#55	Konnte die Passwort-Datei nicht zum Schreiben öffnen.
VNC_NO_SCREENB UFFER	16#56	Nicht genügend Speicher für die Zuweisung des Pixelpuffers.

1.8.4 Befehle der Befehlszeilen-Schnittstelle

Der VNC-Client-Dienst kann mit dem CLI-Befehl vncsvr manuell gestartet oder gestoppt werden:

Syntax

```
vncsvr start | stop | setpass | version
```

Beispiele

```
vncsvr start [watch]
```

VNC-Serverdienst starten

Optional kann "watch" eingegeben werden, um in einem Intervall von 20 Sekunden zu prüfen, ob der VNC-Server noch eingehende Verbindungen annehmen kann.

```
vncsvr stop
```

Stoppt den VNC-Serverdienst (alle Clients werden getrennt!)

```
vncsvr version
```

Zeigt die DLL-Version und die VNC-Protokoll-Versionsnummern des VNC-Servers

```
vncsvr setpass <oldpass> <newpass>
```

Verwenden Sie diesen Befehl, um ein neues VNC-Server-Passwort festzulegen

1.8.5 Konfigurieren des VNC-Servers mit Umgebungsvariablen

Der VNC-Server kann über Umgebungsvariablen konfiguriert werden. Sie können sie manuell setzen, indem Sie jeden Befehl in der CLI kurz vor dem Start des VNC eingeben, oder bequem in der autoexec.lsl definieren.

1.8.5.1 VNC_SVR_ACCEPT_KEYBOARD

Mit dieser Variable wird der Server so eingestellt, dass er Remote-Tastaturereignisse akzeptiert; wenn sie auf TRUE gesetzt ist, akzeptiert der Server Remote-Tastaturereignisse. Bei FALSE werden entfernte Tastaturereignisse abgewiesen.

Defaultwert: TRUE

Siehe auch: [VNCSVRAcceptKeyboardEvents\(\)](#)

Beispiel

```
SETENV VNC_SVR_ACCEPT_KEYBOARD TRUE
```

1.8.5.2 VNC_SVR_ACCEPT_POINTER

Mit dieser Variable wird der Server so eingestellt, dass er entfernte Zeiger-Ereignisse (Maus & Touch) akzeptiert; wenn sie auf TRUE gesetzt ist, akzeptiert der Server entfernte Zeiger-Ereignisse. Bei FALSE werden entfernte Zeiger-Ereignisse abgewiesen.

Defaultwert: TRUE

Siehe auch: [VNCSVRAcceptPointerEvents\(\)](#)

Beispiel

```
SETENV VNC_SVR_ACCEPT_POINTER TRUE
```

1.8.5.3 VNC_SVR_ALWAYS_SHARED

Wenn diese Option auf TRUE gesetzt ist, behandelt der Server alle Verbindungsanfragen so, als ob die Option 'shared connection' gesetzt wäre, unabhängig davon, was der Client angibt.

Defaultwert: FALSE

Siehe auch: [VNCSVRAlwaysShared\(\)](#)

Beispiel

```
SETENV VNC_SVR_NEVER_SHARED TRUE
```

1.8.5.4 VNC_SVR_BLACKLIST_LEVEL

Definiert den Zeitpunkt, an dem ein defekter VNC-Client auf die interne Blacklist gesetzt wird.

- | | |
|---|-----------------------------------------------|
| 0 | Nie, schwarze Liste ist deaktiviert |
| 1 | Test während der Authentifizierung (Standard) |
| 2 | Test beim Verbindungsaufbau |

Beispiel

```
SETENV VNC_SVR_BLACKLIST_LEVEL 1
```

1.8.5.5 VNC_SVR_COMPARE_FB

Diese Eigenschaft legt fest, ob der Server einen Pixelvergleich auf dem Bildpuffer durchführt, um unnötige Aktualisierungen zu reduzieren (TRUE) oder nicht (FALSE).

Defaultwert: FALSE

Siehe auch: [VNCVRCompareFB\(\)](#)

Beispiel

```
SETENV VNC_SVR_COMPARE_FB FALSE
```

1.8.5.6 VNC_SVR_DESKTOP_NAME

Der Desktop-Name des VNC Servers kann über diese Umgebungsvariable definiert werden. Es wird in der Titelleiste des Fensters der VNC-Clients angezeigt. Die maximale Länge des Namens beträgt 64 Zeichen und Abstände sind erlaubt.

Defaultwert: Sigmatek GmbH

Beispiel

```
SETENV VNC_SVR_DESKTOP_NAME Name of the PLC
```

Anforderungen

Die Umgebungsvariable wird ab der Version 01.01.030 der VNC_Server DLM unterstützt.

1.8.5.7 VNC_SVR_DISCONNECT_ON_NONSHARED

Diese Variable definiert das Verhalten des VNC-Servers, wenn ein Client verbunden ist. Ein VNC-Client gibt an, ob er eine Serververbindung mit anderen Clients teilen möchte (geteilte Verbindung), oder nicht (nicht geteilte Verbindung). Je nachdem, wie der Client definiert ist, wird die Verbindung zugelassen oder abgelehnt.

Defaultwert: TRUE

Siehe auch: [VNCSVRCDisconnectClientsOnNonsharedConnection\(\)](#)

Beispiel

```
SETENV VNC_SVR_DISCONNECT_ON_NONSHARED TRUE
```

1.8.5.8 VNC_SVR_DISABLE_SHARED_MEMORY

Diese Umgebungsvariable kann die Vollbildaktualisierung aus dem gemeinsamen Speicher verhindern, wenn sich mehr als 75 % des Bildschirminhalts geändert haben.

Defaultwert: FALSE

Beispiel

```
SETENV VNC_SVR_DISABLE_SHARED_MEMORY TRUE
```

Anforderungen

Diese Funktion wird ab der Schnittstellenversion 01.01.037 unterstützt.

Diese Funktionalität wird nur unter Salamander unterstützt.

1.8.5.9 VNC_SVR_FILTER

Diese Umgebungsvariable wird verwendet, um den Filterstring zu setzen. Für eine detaillierte Beschreibung siehe [VNCSVRCSetFilter\(\)](#).

Defaultwert: +

Beispiel

```
SETENV VNC_SVR_FILTER +192.168.10.30,-
```

1.8.5.10 VNC_SVR_FORCE_BPP

Legt fest, welche Farbtiefe vom VNC-Server an den VNC-Client weitergegeben wird.

0 16-bit Farbtiefe (Default)

1 8-bit Farbtiefe

2 16-bit Farbtiefe

Beispiel

```
SETENV VNC_SVR_FORCE_BPP 1
```

1.8.5.11 VNC_SVR_FULL_UPDATE

Mit dieser Option können Sie den VNC-Server zwingen, nach dem festgelegten Zeitintervall eine Vollbildaktualisierung zu senden. Das Zeitintervall wird in Sekunden angegeben.

Defaultwert: 2

Beispiel

```
SETENV VNC_SVR_FULL_UPDATE 4
```

1.8.5.12 VNC_SVR_LOGFILE

Wenn eine zusätzliche Protokolldatei für die Meldungen des VNC-Servers gewünscht wird, ist folgendes erforderlich:

- Der Logging-Mechanismus muss zunächst angewiesen werden, eine zusätzliche Log-Datei zu schreiben, indem der Log-String mit VNC_SVR_LOGSTR auf *:file:30 festgelegt wird (ein geeigneter Debug-Link muss gewählt werden).
- Der Dateiname muss dann mit VNC_SVR_LOGFILE angegeben werden.

Defaultwert: C:\vncsvr.log

Beispiel

```
SETENV VNC_SVR_LOGFILE c:\vncsvr.log
```

1.8.5.13 VNC_SVR_LOGSTR

Mit dieser Umgebungsvariable wird der Log-String definiert. Für eine genauere Beschreibung, siehe Log-String Beschreibung.

Default-Wert: *:kernel_svr:0

Beispiel

```
SETENV VNC_SVR_LOGSTR *:kernel_svr:30
```

1.8.5.14 VNC_SVR_MAX_CONNECTIONS

Definiert, wie viele gleichzeitige Verbindungen der Server erlaubt.

Defaultwert: 2 mit Systemen gleich oder kleiner 256 MB RAM, 4 bei Systemen mit mehr als 256 MB RAM

Unter Salamander ist die Anzahl auf 6 begrenzt.



Siehe auch: [VNCSVRMMaxConnections](#)

Beispiel

```
SETENV VNC_SVR_MAX_CONNECTIONS 2
```

1.8.5.15 VNC_SVR_NEVER_SHARED

Wenn diese Option auf TRUE gesetzt ist, gibt der Server eine Verbindung nicht frei, unabhängig davon, was der Client angibt.

Defaultwert: FALSE

Siehe auch: [VNCSVRNeverShared\(\)](#)

Beispiel

```
SETENV VNC_SVR_NEVER_SHARED TRUE
```

1.8.5.16 VNC_SVR_PORT

Definiert die Port-Nummer, die der VNC-Server verwenden sollte.

Defaultwert: 5900

Beispiel

```
SETENV VNC_SVR_PORT 5900
```

1.8.5.17 VNC_SVR_PORT2

Definiert eine zweite Port-Nummer, die vom VNC-Server verwendet wird.

Defaultwert: Standardmäßig verwendet der VNC-Server nur die in [VNC_SVR_PORT](#) angegebene Port-Nummer.

Beispiel

```
SETENV VNC_SVR_PORT2 6900
```

1.8.5.18 VNC_SVR_TIMEOUT_INACTIVE

Legt die Zeit in Sekunden fest, nach der der VNC-Server die Verbindung eines inaktiven Clients trennt.

Defaultwert: 3600

Beispiel

```
SETENV VNC_SVR_TIMEOUT_INACTIVE 3600
```

1.8.5.19 VNC_SVR_WAIT_FOR_CLIENT

Diese Eigenschaft legt fest, wie lange (in Millisekunden) der Server auf einen Client wartet, der nicht antwortet.

Defaultwert: 20000

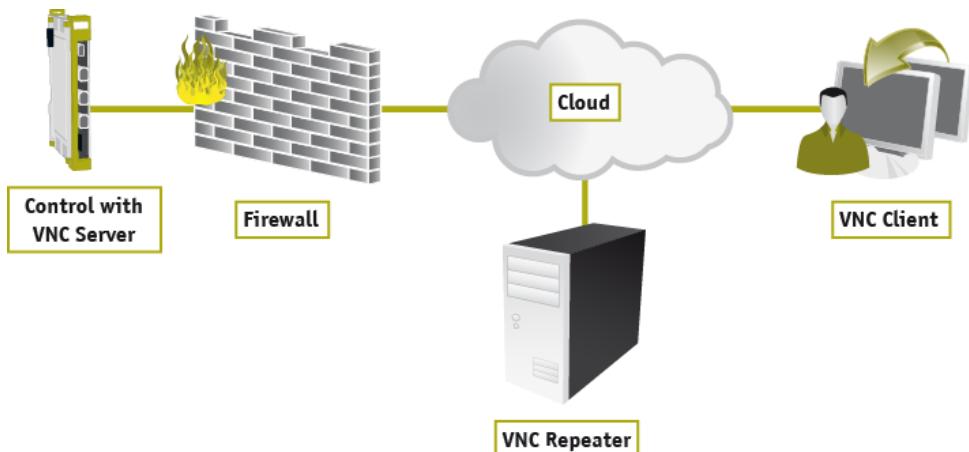
Siehe auch: [VNCVRClientWaitTimeMillis\(\)](#)

Beispiel

```
SETENV VNC_SVR_WAIT_FOR_CLIENT 50000
```

1.8.6 Repeater-Funktion

Die VNC-Server-DLM für SIGMATEK-Steuerungen haben eine Repeater-Erweiterung. Damit wird die Verbindung zu VNC-Servern (Steuerungen) hergestellt, die im Netzwerk nicht erreichbar sind (z. B. hinter einer Firewall liegen). Steuerung mit dem VNC-Server und VNC-Clients über eine Verbindung zu einem Repeater, der den Datenaustausch durchführt.



Damit der Repeater gestartet wird, muss eine gültige vncsvr.cfg vorhanden sein. Die Erweiterung ist ab der Version 1.1.020 verfügbar.

1.8.6.1 Starten der Repeater-Funktion

Der DLM wird wie üblich mit vncsvr start in der CLI oder Autoexec.lsl gestartet. Die Repeater-Funktion wird gestartet, wenn eine gültige vncsvr.cfg vorhanden ist; weitere Eingaben sind nicht erforderlich.

Vor der Verwendung muss ein Passwort eingegeben werden.

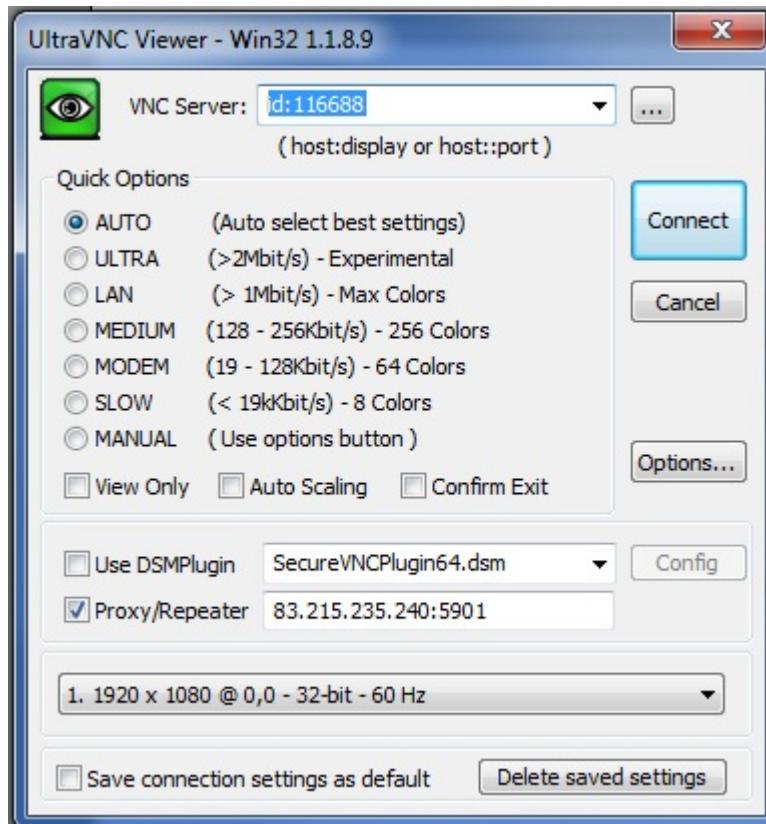
```
vncsvr setpass Sigmatek mypass.
```

Unter bestimmten Bedingungen ist die Einrichtung eines Gateways erforderlich (fragen Sie den Netzwerkadministrator nach diesen Informationen).

```
set ip gateway 192.168.1.1
```

1.8.6.2 Verwenden der Repeater-Funktion

Jeder VNC-Client kann als Client verwendet werden, in dem ein Repeater (Proxy) eingetragen ist. Hier wird Ultra VNC als Beispiel verwendet.



Als VNC-Server wird die ID aus vncsvr.cfg verwendet. Hier muss "id" vor die eigentliche ID geschrieben werden. Unter Proxy/Repeater werden die IP aus vncsvr.cfg und der Port (5901) eingetragen. Das Passwort wird dann vor dem Verbindungsaufbau abgefragt.

1.8.6.3 Konfigurieren der Repeater-Funktion

Die Konfigurationsdatei vncsvr.cfg, bei der es sich um eine Textdatei handelt, die im Ordner C:\LSLSYS gespeichert sein muss, startet die Repeater-Funktion. Um gültig zu sein, muss diese Datei die folgenden Schlüssel-Wert-Paare enthalten.

```
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5500
REPEATER_ID = $SERIALNUM
```

Die REPEATER-IP enthält die IP und REPEATER-PORT, den Port des Repeaters, der verwendet werden soll. DNS kann auch als IP definiert werden.

Unter REPEATER ID kann keine individuelle ID vergeben werden. Diese ID muss numerisch sein und darf maximal acht Zeichen lang sein. Wenn als Wert \$SERIALNUM eingegeben wird, meldet sich der VNC-Server mit der Seriennummer der Steuerung am Repeater an. Der Kommentar kann mit einer Raute "#" gesetzt werden.

1.8.7 Beispielkonfigurationen mittels AUTOEXEC.LSL

1.8.7.1 Szenario 1: Der Server sollte genau eine Verbindung von einem bestimmten Host zulassen

Im ersten Szenario sollte der VNC-Server so konfiguriert werden, dass er nur eine Verbindung von einer bestimmten IP-Adresse (z. B. 192.168.30.45) annimmt und Remote-Tastatur- und Touch/Maus-Eingaben akzeptiert. Der VNC-Dienst sollte Port 6300 scannen und inaktive Clients nach 2 Stunden (7200 Sekunden) trennen.

- Stellen Sie zuerst sicher, dass der VNC-Server korrekt [installiert](#) ist.
- Bearbeiten Sie die AUTOEXEC.LSL und fügen Sie die folgenden Regeln hinzu:

```
REM ##### VNC-SERVER-CONFIGURATION #####
SETENV  VNC_SVR_PORT          6300
SETENV  VNC_SVR_FILTER         +192.168.30.45,-
SETENV  VNC_SVR_ACCEPT_POINTER TRUE
SETENV  VNC_SVR_ACCEPT_KEYBOARD TRUE
SETENV  VNC_SVR_MAX_CONNECTIONS 1
SETENV  VNC_SVR_TIMEOUT_INACTIVE 7200

REM ##### START VNC-SERVER #####
VNCsvr START
```

1.8.7.2 Szenario 2: Der Server soll eine Verbindung mit jedem Host erlauben, aber nur zur Anzeige

Im zweiten Szenario sollte der VNC-Server so konfiguriert werden, dass er jede eingehende Verbindungsanfrage bis zu 5 verbundenen Clients akzeptiert, aber keine Remote-Eingabe. Der VNC-Dienst sollte den Standard-Port 5900 scannen und inaktive Clients nach 8 Stunden (28800 Sekunden) trennen.

- Stellen Sie zuerst sicher, dass der VNC-Server korrekt installiert ist.
- Bearbeiten Sie die AUTOEXEC.LSL und fügen Sie die folgenden Regeln hinzu:

```
REM ##### VNC-SERVER-CONFIGURATION #####
SETENV VNC_SVR_FILTER      +
SETENV VNC_SVR_ACCEPT_POINTER FALSE
SETENV VNC_SVR_ACCEPT_KEYBOARD FALSE
SETENV VNC_SVR_MAX_CONNECTIONS 5
SETENV VNC_SVR_TIMEOUT_INACTIVE 28800

REM ##### START VNC-SERVER #####
VNCSVRF START
```

1.8.7.3 Szenario 3: Einrichten des Servers unter Verwendung einer separaten Protokolldatei

Im zweiten Drittel sollte der VNC-Server so konfiguriert werden, dass er jede eingehende Verbindungsanfrage von allen IPs aus einem bestimmten Subnetz (192.168.30.0) bis zu 4 verbundenen Clients und Remote-Eingang akzeptiert. Der VNC-Server sollte den Standard-Port 5900 scannen und seine Protokolldaten in die Datei c:\vncsvr.log schreiben, um eine detaillierte Protokolldatei (einschließlich Debug-Meldungen) zu erhalten.

- Stellen Sie zuerst sicher, dass der VNC-Server korrekt installiert ist.
- Bearbeiten Sie die AUTOEXEC.LSL und fügen Sie die folgenden Regeln hinzu:

```
REM ##### VNC-SERVER-CONFIGURATION #####
SETENV VNC_SVR_FILTER      +192.168.30.0/255.255.255.0,-
SETENV VNC_SVR_LOGSTR      *:file:100
SETENV VNC_SVR_LOGFILE     c:\vncsvr.log
SETENV VNC_SVR_ACCEPT_POINTER TRUE
SETENV VNC_SVR_ACCEPT_KEYBOARD TRUE
SETENV VNC_SVR_MAX_CONNECTIONS 4

REM ##### START VNC-SERVER #####
VNCSVRF START
```

1.8.7.4 Szenario 4: Einen Server einstellen, der Remote-Events in das Benutzerprotokoll event02.log schreibt

Im vierten Szenario sollen alle Remote-Befehle (Tastatur- und Mouse/Touch-Ereignisse) von den VNC-Clients in der Benutzerprotokolldatei event02.log protokolliert werden. Die folgenden Zeilen müssen daher in der autoexec.ls1 hinzugefügt werden:

```
REM ##### TURN ON EVENT - LOGGING #####
SET USERLOG2 ON
REM #####
```

1.8.8 Beispielkonfiguration mit der Datei vncsvr.cfg

Die Datei vncsvr.cfg wird für den Zugriff auf den VNC-Server über einen Repeater benötigt (siehe [Repeater](#)).

1.8.8.1 Szenario 1: VNC-Server meldet sich mit der Seriennummer an

Im ersten Szenario meldet sich der VNC-Server mit der Seriennummer am Repeater an. Die folgenden Zeilen müssen daher in der vncsvr.cfg hinzugefügt werden:

```
##### Configuration - Repeater Logged-On with Serial number#####
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5500
REPEATER_ID = $SERIALNUM
```

1.8.8.2 Szenario 2: VNC-Server meldet sich mit einer individuellen ID an

Im zweiten Szenario meldet sich der VNC-Server mit einer individuellen ID (113355) beim Repeater an. Die folgenden Zeilen müssen daher in der vncsvr.cfg hinzugefügt werden:

```
##### Configuration - Repeater Logged-On with individual ID #####
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5500
REPEATER_ID = 113355
```

1.8.8.3 Szenario 3: VNC-Server meldet sich mit DNS an

Im dritten Szenario meldet sich der VNC-Server mit einem Namen (DNS) beim Repeater an. Die folgenden Zeilen müssen daher in der vncsvr.cfg hinzugefügt werden:

```
##### Configuration - Repeater Logged-On with DNS #####
REPEATER_IP = Sigmatek.at/repeater
REPEATER_PORT = 5500
REPEATER_ID = $SERIALNUM
```

1.9 SRAM

1.9.1 Formate

Mit der Umgebungsvariable SRAMFORMAT (SETENV SRAMFORMAT 1|2) wird eingestellt, welches SRAM-Format verwendet werden soll.

Das SRAMFORMAT hat nichts mit der Einstellung SET RAM FORMAT NEWW zu tun. Diese Einstellung ist obsolet und muss nicht verwendet werden. Standardmäßig wird bei den Plattformen C-IPC, TEACHBOX und Edge2 das Format 2 verwendet, bei den übrigen Plattformen das Format 1.

Wenn ein SRAM-Format mit "SETENV SRAMFORMAT" eingestellt wird, sollte dieser Befehl auch in der Autoexec.lsl enthalten sein.

1.9.1.1 Format 1

Daten- und Steuerstrukturen der RAM- und RAMEx-Objekte befinden sich im SRAM.

Dateien:

C:\LSDL DATA\SRAM.CPY ... temporäres Backup für den SRAM-Reorg wird beim Hochfahren verwendet.

Da bei diesem Format die Kontrollstrukturen im SRAM liegen und der SRAM-Speicher eine gewisse Fragmentierung aufweist, ist der in der LASAL-Klasse im Projekt-Info-Fenster angezeigte Wert nur ein Näherungswert, d.h. Sie können aus diesem Wert nicht genau ermitteln, wie viele RAM- bzw. RAMEX-Objekte noch angelegt werden können.

1.9.1.2 Format 2

Dieses Format kann nur auf Plattformen mit einem Dateisystem verwendet werden.

Statische Kontrollstrukturen der RAM- und RAMEx-Objekte befinden sich in einer Datei, die Daten liegen im SRAM. Die statischen Kontrollstrukturen verändern den Start der Anwendung, wenn RAM/RAMEx-Objekte hinzugefügt oder entfernt werden.

Der Vorteil dieser Formate ist, dass das gesamte SRAM für Daten zur Verfügung steht und somit mehr Daten gespeichert werden können.

Dateien:

C:\RAMFILE.DAT ... statische Kontrollstrukturen

C:\LSDL DATA\RAMFILE.CPY

C:\LSDL DATA\SRAM.CPY ... temporäres Backup für den SRAM-Reorg wird beim Hochfahren verwendet.

1.9.1.3 Konvertieren zwischen Formaten

Ab der Loader-Version 02.02.123 können SRAM-Daten zwischen den Formaten konvertiert werden.

Die Daten werden beim Starten der Anwendung automatisch konvertiert. Das Format des tatsächlichen SRAM-Inhalts wird mit der Einstellung in der Umgebungsvariable SRAMFORMAT verglichen. Wenn die Formate unterschiedlich sind, werden die Daten konvertiert.

Es ist zu beachten, dass eine Konvertierung von Format 2 nach Format 1 nur dann möglich ist, wenn im SRAM genügend Platz für Daten im Format 1 vorhanden ist, da in diesem Format die SRAM-Daten deutlich mehr Speicherplatz belegen.

Wenn eine Konvertierung nicht möglich ist, bleiben die Daten im Format 2.

1.9.2 Remanente Server

Ein remanenter Server kann einen 4-Byte-Wert beibehalten.

Wenn beim SRAM-Server eingestellt, werden die Daten im SRAM gespeichert. Mit der Einstellung FILE werden die Daten in der Datei C:\RETSVR.DAT gespeichert.

Wurde die Einstellung von SRAM auf FILE oder umgekehrt geändert, so werden ab der Version 02.02.123 die Daten vom anderen Medium übernommen.

Ab Version 02.02.123 kann mit der Funktion LDR_IsAsnycFileOperationInProgress ermittelt werden, ob asynchrone Dateioperationen noch aktiv sind (Retentive File). Diese Funktion wird benötigt, wenn auf mehrere remanente Server geschrieben wird (z.B. beim Laden eines Werkzeugkatalogs) und anschließend ein Neustart durchgeführt wird.

Da die Daten beim Schreiben auf einen remanenten Server asynchron zum Anwendungsprogramm in der Sicherungsdatei gespeichert werden, ist ein Neustart nur möglich, wenn die Daten im remanenten Server gesichert sind.

1.9.3 RamEx

Ab RamEx Version 1.6, MerkerEx Version 1.2, StringInternal Version 1.5 und Loader Version 02.02.123 können RamEx-Daten in der jeweiligen Datei gespeichert werden. Der UseFile-Client muss dabei auf 1 gesetzt werden - dies ist nur mit InitValue in LASAL möglich.

Die Dateien werden im Verzeichnis C:\LSLDATA\RAMEX gespeichert. Zusätzlich wird im Verzeichnis C:\LSLDATA eine Indexdatei RAMEX.IDX abgelegt, in der die gültigen RamEx-Dateien eingetragen sind.

Wenn ein RamEx-Objekt später so konfiguriert wird, dass die Daten in einer Datei gespeichert werden und bereits SRAM-Daten vorhanden sind, werden die Daten aus dem SRAM übernommen und in der Datei gespeichert. Der SRAM-Eintrag wird dann beim Reorganisieren gelöscht.

Die Daten werden auch übernommen, wenn ein RamEx-Objekt von Datei auf SRAM umkonfiguriert wird. Dann wird die Datei gelöscht.

1.9.4 Speichern, wiederherstellen, löschen

1.9.4.1 SRAM sichern

Mit dem CLI-Befehl [SRAMSAVE](#), können die SRAM-Daten in einer Datei gesichert werden.

SRAMSAVE <filename>

Sichert den SRAM-Inhalt in einer Datei.

SRAMLOAD <filename>

Stellt den SRAM-Inhalt aus einer Sicherungsdatei wieder her.

Ab der OS-Version 01.02.195 werden auch remanente Dateidaten und RamEx-Dateien gesichert. Es ist weiterhin möglich, alte Sicherungsdateien (ohne RETSVR.Dat und RamEx-Dateien) zu laden.

Beim Sichern und Wiederherstellen von Daten mit dem CLI-Befehl [SRAMSAVE](#) kann das Format nicht konvertiert werden. Das bedeutet, dass es nicht möglich ist, ein im Format 2 gespeichertes SRAM mit Format 1 zu sichern.

1.9.4.2 SRAM löschen

SRAMCLEAR

Löscht Restdaten, die im SRAM (RAM, RAMEx, remanenter SRAM-Server) gespeichert sind.

SRAMCLEARFILE

Löscht Restdaten, die in der Datei (Dateiserver) gesichert wurden.

SRAMCLEARALL

Löscht alle Datenreste (sowohl im SRAM als auch in der Datei).

1.9.4.3 Laden einer Reorganisationskopie

Ab der Loader-Version 02.02.123 kann eine SRAM-Kopie, die bei der Reorganisation des SRAMs erstellt wurde, wieder geladen werden.

Die *.CPY-Dateien im Verzeichnis C:\LSLDATA müssen in *.INI umbenannt werden. Bei SRAMFORMAT 1 wird dann die Datei SRAM.INI benötigt - bei Format 2 werden die Dateien SRAM.INI und RAMFILE.INI benötigt.

Das SRAM-Format muss mit den INI-Dateien übereinstimmen. Nach dem Laden der Dateien werden diese in .DON (DONE) umbenannt.

Diese Dateien werden beim Reorganisieren (nach Neustart) des SRAMs geladen, wenn sie im Verzeichnis LSLDATA vorhanden sind und das SRAM gültig ist (nicht CPU-Status "Ldr out of near" oder ab OS-Version 01.03.090 "SRAM-Error").

Bei einer neuen CPU (die Datei isnotnew.inf ist NICHT vorhanden) werden die Dateien auch geladen, wenn das SRAM ungültig ist (nur beim ersten RUN nach PowerON).

Wenn z.B. die CF-Karte eines C-IPC neu eingelegt wird und die SRAM-Daten (Version vom letzten Neustart) geladen werden sollen, müssen die *.CPY-Dateien im Verzeichnis LSLDATA in *.INI umbenannt und die Datei isnotnew.inf gelöscht werden.

1.9.4.4 LASAL CLASS Tool Ram Image

Auch mit dem LASAL CLASS-Tool RAM Image können SRAM-Daten gesichert, wiederhergestellt und gelöscht werden. Die Funktionalität dieses Werkzeugs ist in der LASAL-Online-Hilfe beschrieben.

1.9.5 SRAMDisk

Bei der Edge-Plattform werden die nullspannungssicheren Daten im RAM gespeichert und (oder nur) im Hintergrund zyklisch beim Herunterfahren auf einen reservierten Platz auf der SD-Karte (=SRAMDisk) gesichert.

Das Verhalten der SRAM-Datensicherung kann mit den folgenden Parametern eingestellt werden. Die Werte können mit dem CLI-Befehl SET SRAMDISK oder der LASAL-CLI-Schnittstelle "SRAMDISK" abgefragt oder geändert werden.

Parametername	Bedeutung
NPD (Default 32)	N-PowerDownSectors. Die Anzahl der Sektoren, die während des Abschaltens geschrieben werden kann. Die Größe eines Sektors beträgt 512 Byte.
T1 (Default 3)	T1 bestimmt, wie schnell die Anzahl der geänderten Sektoren innerhalb einer bestimmten Zeit auf einen Wert < NPD gesetzt werden soll. (Einheit: Sekunden, -1 = Wert nicht aktiv)

T2 (Default 30)	T2 bestimmt die Zeit, nach der die geänderten Sektoren auf die Platte geschrieben werden, auch wenn die Anzahl der geänderten Sektoren bei einem Wert < NPD liegt. (Einheit: Sekunden, -1 = Wert nicht aktiv)
T3 (Default 60)	T3 bestimmt, wie oft eine konsistente Version auf dem Datenträger erstellt und innerhalb einer bestimmten Zeit auf eine neue Versionsnummer geändert wird. (Einheit: Sekunden, -1 = Wert nicht aktiv)
FBFW (Default 1)	Flush Before File Write. Wenn FBFW auf dem Wert ≠ 0 steht, werden alle geänderten SRAM-Daten auf die Festplatte geschrieben, bevor in eine Datei geschrieben wird.
ONLYPD (Default 0)	Wenn ONLYPD auf einem Wert > 0 steht, werden im laufenden Betrieb (nur bei PowerDown) keine SRAM-Daten auf die Platte geschrieben. Im Unterschied zum Parameter SramdiskFullCopyAtPowerdown (siehe Beschreibung unten) werden hier SRAM-Schreibvorgänge weiterhin von der MMU erkannt, was die CPU-Last erhöht. Wenn also auf eine zyklische Sicherung der SRAM-Daten im Hintergrund verzichtet wird, ist der Parameter SramdiskFullCopyAtPowerdown vorzuziehen.

In der optionalen Datei C:\LSLSYS\CONFIG.LSL befinden sich die Einstellungen für die SRAMDisk-Größe und das Ausschaltverhalten.

Parametername	Bedeutung
SramSize (Default 524288)	SRAM Größe. Ein durch 4096 teilbarer Wert muss gesetzt werden.
SramdiskFullCopyAtPowerdown (Default 0)	Bei einem Wert von 1 wird das gesamte SRAM auf die Festplatte geschrieben, unabhängig davon, ob sich der SRAM-Inhalt geändert hat oder nicht. Im Hintergrund werden keine SRAM-Daten auf die Festplatte geschrieben.

Seit OS Version 01.02.195.

Parametername	Bedeutung
SysSramSize	Größe des SRAM-Bereichs, der für das Betriebssystem verwendet wird (z. B. für den temporären Puffer mit der Ereignisprotokolldatei). Eine minimale Größe von 8 k wird empfohlen. Wenn der Wert zu klein ist, kann eine Protokollmeldung in der Ereignisprotokolldatei verloren gehen. Es ist wichtig zu beachten, dass die Änderung dieses Parameters auch die SRAM-Größe für die Anwendung ändert. Das heißt, wenn sysSramSize erhöht wird, hat die Anwendung weniger SRAM zur Verfügung, was zu SRAM-Datenverlust führen kann. Wenn dieser Parameter nicht konfiguriert wird, wird die Größe folgendermaßen berechnet.

```
if sramSize ≥ 128 k then sysSramSize = 16 k
else if sramSize ≥ 32 k then sysSramSize = sramSize / 8
else sysSramSize = sramSize / 16
```

Mit "scramdiskFullCopyAtPowerdown = 1" und einem sramSize-Wert von 16 kByte kann die Schreibrate auf der SD-Karte auf ein Minimum reduziert und damit die Lebensdauer der Karte verlängert werden. Von diesen 16 kByte benötigt das Betriebssystem 4 kByte

zum sicheren Schreiben von Ereignisprotokolldateien. Für den Anwender verbleiben somit 12 kByte; dies entspricht ca. 3000 SRAM-Werten.

Ab OS-Version 01.02.195 bleiben die Daten bei einer Größenänderung so lange erhalten, wie bei einer Verkleinerung des Datenträgers genügend Platz vorhanden ist.

Ab der Betriebssystemversion 01.02.195 werden Daten auch dann auf die SRAMDisk geschrieben, wenn das SRAM so konfiguriert ist, dass die Daten nur beim PowerDown geschrieben werden (ONLYPD oder SramdiskFullCopyAtPowerdown).

1.9.5.1 SRAMDisk Backup-Mechanismen

Es wird unterschieden zwischen Sicherung im laufenden Betrieb und Sicherung nur beim Ausschalten.

In beiden Fällen gibt es zwei Bereiche auf der SD-Karte, wobei einer dieser Bereiche immer ein fertiges konsistentes Abbild des SRAM enthält und die neu zu sichernden Daten in den anderen Bereich geschrieben werden.

1.9.5.1.1 Sichern von Daten während des Betriebs und beim Ausschalten

Remanente Daten liegen im RAM und werden nach einer Änderung im Betrieb im Hintergrund auf die SD-Karte geschrieben. Beim Ausschalten werden die restlichen, bis dahin nicht geschriebenen Daten auf die SD-Karte geschrieben.

Die remanenten Daten im RAM sind seitenweise durch die MMU (Memory Management Unit der CPU) schreibgeschützt. Sobald der Anwender in einen Bereich schreibt, kommt es zu einer Exception im Betriebssystem. Die Größe einer Seite beträgt 4k, in der Ausnahme ist die Adresse, auf die geschrieben wird, verfügbar, so dass das Betriebssystem weiß, wenn sich Daten im remanenten Bereich geändert haben. Dann wird der Schreibschutz der Seite aufgehoben.

Eine 4k-Seite wird weiter in 8x 512-Byte-Sektoren aufgeteilt, was die minimale Schreibgröße einer SD-Karte ist.

Zyklisch werden die Sektoren einer entsperrten Seite überprüft, ob sich der Inhalt im Vergleich zu einer Sicherungsseite, die den Inhalt der SD-Karte spiegelt, geändert hat. Wenn dies der Fall ist, wird der Sektor als geändert markiert. Geänderte Sektoren werden dann im Hintergrund auf die SD-Karte geschrieben. Sobald alle Sektoren einer Seite auf die SD-Karte geschrieben wurden, wird der Schreibschutz der Seite wieder aktiviert. Die Häufigkeit des Schreibens von geänderten Sektoren auf die SD-Karte wird durch die Parameter NPD, T1, T2 und T3 gesteuert.

Änderungen der remanenten Daten führen also zu Schreibvorgängen auf der SD-Karte, wodurch sich die Lebensdauer der SD-Karte verkürzt. Je mehr und je häufiger remanente Daten geändert werden, desto mehr Schreibvorgänge auf der SD-Karte finden statt.

Auch die Verteilung der Daten im SRAM spielt hier eine Rolle. Je näher die Daten nebeneinander liegen, desto weniger Sektoren sind von Änderungen betroffen und desto weniger Schreibvorgänge auf der SD-Karte sind erforderlich. Der Anwender kann aber keinen Einfluss auf die Verteilung der Daten nehmen.

1.9.5.1.2 Sichern von Daten nur beim Ausschalten

Remanente Daten befinden sich im RAM und werden erst beim Ausschalten auf die SD-Karte geschrieben.

So verursachen Änderungen von remanenten Daten keine Schreibvorgänge auf die SD-Karte zur Laufzeit und keine höhere CPU-Last durch Hintergrundverarbeitung.

In dieser Einstellung kann die Größe der remanenten Daten also nur so hoch sein, dass die Zeit für die Sicherung beim Ausschalten lang genug ist. Der Wert beträgt hier 16 kByte. Von diesen 16 kByte benötigt das Betriebssystem 4 kByte zum sicheren Schreiben von Ereignisprotokolldateien. Für den Anwender verbleiben somit 12 kByte; dies entspricht ca. 3000 SRAM-Werten.

1.9.5.1.3 Vorteile und Nachteile dieser beiden Arten der Datensicherung

- Datenverlust, wenn das Schreiben auf die SD-Karte beim Ausschalten nicht funktioniert
 - Werden die Daten während des Betriebs gesichert, ist ein maximal einige Minuten alter Datensatz noch verfügbar. Wenn Daten nur beim Ausschalten gespeichert werden, sind die letzten verfügbaren Daten die des Einschaltens.
- Belastung der SD-Karte durch Schreibvorgänge
 - Beim Sichern von Daten während des Betriebs ist die Anzahl der Schreibvorgänge auf der SD-Karte höher. Die Anzahl ist abhängig von der Anzahl der SRAM-Änderungen, der Verteilung der SRAM-Daten und den verwendeten Einstellungen.
- CPU-Auslastung
 - Beim Sichern von Daten im laufenden Betrieb ist die CPU-Belastung durch das Sichern im Hintergrund und das Freigeben einer Seite zum Schreiben in das SRAM höher.

1.9.5.1.4 Überwachung der Anzahl von Schreibvorgängen auf der SD-Karte

Die Anzahl der Sektor-Schreibvorgänge auf die SD-Karte wird vom Betriebssystem gezählt und protokolliert. Die aktuellen Zählerstände sind auch über Systemvariablen zugänglich.

Die Zählerstände werden alle 10 Minuten in die Datei c:\sysmsg\<volume-serial>.wst geschrieben, wobei \<volume-serial> die Seriennummer der Partition der SD-Karte ist. Der Inhalt der Datei hat den folgenden Aufbau:

```
<N1> writes, <N2> seconds, <N3> sramdisk-writes
```

N1	Gesamtzahl der Schreibvorgänge des Sektors
N2	Zeit, wie lange die SD-Karte bereits in Gebrauch ist (in Sekunden)
N3	Anzahl der Sektor-Schreibvorgänge aufgrund von SRAM-Disk-Sicherungen

Mit den folgenden Systemvariablen können Sie auf diese Werte zugreifen:

_diskOperatingSeconds	Zeit, wie lange die SD-Karte bereits in Gebrauch ist (in Sekunden)
_diskSectWrCntr	Gesamtzahl der Schreibvorgänge des Sektors
_diskSectWrCntr1000Sec	Anzahl der Sektor-Schreibvorgänge in den letzten 1000 Sekunden
_diskSectWrCntr10Sec	Anzahl der Sektor-Schreibvorgänge in den letzten 10 Sekunden
_diskSectWrCntrSramdisk	Anzahl der Sektor-Schreibvorgänge aufgrund von SRAM-Disk-Sicherungen

1.9.5.2 Erkennen eines Fehlers beim Schreiben auf die SRAMDisk während des Herunterfahrens

Ab der Betriebssystemversion 01.03.011 wird angegeben, ob ein Schreiben auf die SRAMDisk während des Power-Downs möglich ist oder nicht. Ab OS-Version 01.03.070 wird diese Information nur noch bereitgestellt, wenn die zyklische Sicherung (sramdiskFullCopyAtPowerdown = 1) nicht aktiviert ist.

Diese Informationen werden vom Loader ab der Version 02.02.158 ausgewertet und im Fehlerfall wird im Loader ein SRAM-Fehler ausgelöst (die SRAM-Applikationsdaten werden dabei zurückgesetzt).

Diese Rückmeldung des Loaders kann ab der Loader-Version 2.2.167 deaktiviert werden:

Wenn die Umgebungsvariable NO_SRAM_POWERDOWN_CHECK existiert und einen Wert ungleich "0" hat, dann wird der Test zur Überprüfung, ob das SRAM während des Power-Downs beschrieben wurde, übersprungen.

Beispiel

```
SETENV NO_SRAM_POWERDOWN_CHECK 1
```

1.9.6 Reaktion, wenn ein Fehler im SRAM erkannt wird

Wenn der Loader einen Fehler in der SRAM-Datenstruktur feststellt, wird das gesamte SRAM auf 0 oder auf den Initialwert gesetzt und der Applikationsstart angehalten (CPU-Status Ldr.Out-Of-Near oder ab OS-Version 01.03.090 "SRAM-Error"). Die Steuerung muss dann von Hand neu gestartet werden. Dadurch wird sichergestellt, dass die Steuerung nicht unbemerkt mit den Initialisierungswerten startet.

Wenn im Fehlerfall nur das SRAM auf die Init-Werte gesetzt wird und die Applikation starten soll, muss die Umgebungsvariable SRAM_CONTINUE_ON_ERROR auf einen Wert ungleich 0 gesetzt werden.

```
SETENV SRAM_CONTINUE_ON_ERROR 1
```

1.9.7 SRAM reorganisieren

Beim ersten Applikationsstart nach dem Einschalten oder beim Projektwechsel wird das SRAM auf Plattformen mit Dateisystem reorganisiert. Der SRAM-Inhalt wird dabei in eine temporäre Kopie geschrieben, das SRAM wird anschließend wieder aufgebaut. Wenn der Wert in der Kopie verfügbar ist, wird er verwendet. Ansonsten wird der im Projekt gesetzte Init-Wert verwendet. Durch die Reorganisation werden die nicht benötigten Remix-Objekte aus dem SRAM entfernt.

Die nicht mehr benötigten RamEx-Dateien werden durch die Reorganisation gelöscht.

Soll die Reorganisation sowohl nach dem Einschalten als auch nach einem späteren "RESET+RUN" durchgeführt werden, muss die Funktion LDR_ForceSramReorgOnNextRun aufgerufen werden.

Der Aufruf dieser Funktion bewirkt die Reorganisation der SRAM mit dem nächsten RUN.

Wenn die Umgebungsvariable SRAM_DISABLE_REORG auf einen Wert ungleich 0 gesetzt ist, wird die Reorganisation des SRAMs unterdrückt.

Dateien

C:\LSLDATA\REORGINF.DAT

In dieser Datei wird der Projektname und eine OS-Instanz-ID gespeichert. Mit diesen Werten kann der Loader feststellen, ob die Reorganisation mit dem ersten Applikationsstart nach dem Einschalten durchgeführt wird oder ein Wechsel in ein anderes Projekt erfolgt ist.

1.9.8 Größe und Speicheraufteilung des SRAMs

Die Größe des physikalisch vorhandenen SRAMs ist im Datenblatt der CPU ersichtlich:

Beispiel CP 731-K: CP 731-K.pdf

Interner Datenspeicher (SRAM) 512 kB (durch Batterie gepuffert)

Davon OS wird ein Teil für das Betriebssystem verwendet. Bei Salamander CPUs sind das üblicherweise 16 kB. Der Rest steht für die Applikation für die nullspannungssicheren Objekte (RAM- und RAMEx-Objekte, retentive Server Sram) zur Verfügung.

Der konkrete Wert kann über die Klasse `_MemoryInformation` oder über die globale Variable `_S_Ram_Hptr^.DataLength` abgefragt werden.

Beispiel CP 731-K

```
_MemoryInformation.SRAMTotal 507 840
```

```
_S_Ram_Hptr^.DataLength 507 840
```

Wie viele RAM- und RAMEx-Objekte dann tatsächlich verwendet werden können, hängt erst einmal vom verwendeten Sram Format ab. Beim Sram Format 1 werden im Sram auch statische Verwaltungsinformation abgelegt. Das wurde mit dem Sram Format 2 geändert, hier wird ein Großteil der statischen Verwaltungsinformation in das File C:\ramfile.dat ausgelagert. Das Sram Format 1 ist veraltet und wird bei Salamander CPUs nicht mehr verwendet. Es wird daher hier nicht näher beschrieben.

Speicheraufteilung im Sram Format 2:

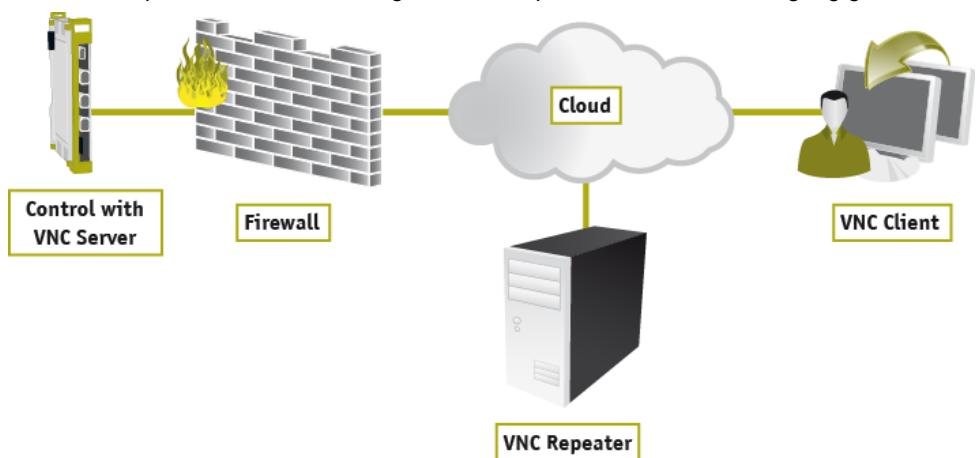
- Der Sram Speicher, welcher der Applikation zur Verfügung steht, wird folgendermaßen aufgeteilt:
- In jedem Fall wird ein 72 Byte Header benötigt
- pro RAM Objekt werden 4 Bytes benötigt
- Ein retentive Server Sram hat dengleiche Speicherbedarf wie ein RAM Objekt
- Pro RAMEx Objekt wird ein 16 Byte Header plus die eigentlichen Daten der RAMEx benötigt

Der für die RAMEx Daten benötigte Speicher wird allerdings nicht byteweise sondern in Einheiten zu je 4 Bytes allokiert. Weiters kann durch häufiges Ändern der Größe der RAMEx Daten der Speicher fragmentiert werden, wodurch ein ungenutzter Sram Speicher entsteht. Beispiel: Wenn 2 RAMEx Objekte mit je 1000 Bytes existieren, und die Größe

des ersten RAMEx Objekts wird auf 800 Bytes reduziert, dann entstehen 200 Bytes ungenutzter SRAM Speicher.

1.10 Repeater

Die LASAL-Repeater-Funktion für SIGMATEK-Steuerungen wurde entwickelt, um sich mit Steuerungen zu verbinden, welche vom Internet aus keine Verbindungen annehmen (zulassen). Steuerungen mit LASAL-Repeater bauen eine Verbindung zu einem Repeater auf. Diese Verbindung wird aufrechterhalten, solange die Repeater-Funktionalität auf der Steuerung aktiv ist. Nachdem sich die Steuerung auf dem Server angemeldet hat, werden die Standardports von der Steuerung auf dem Repeater-Server zur Verfügung gestellt.



1.10.1 LASAL OS mit Repeater-Funktionalität

1.10.1.1 Konfiguration

Damit der LASAL-Repeater gestartet werden kann, muss eine gültige `lslrepeater.cfg` unter `C:\LSSLSSYS\` vorhanden sein. Das Anmelden mit der jeweiligen ID und Passwort muss zudem auf dem Repeater-Server hinterlegt sein, da dieser eine White-List verwaltet, die alle gültigen IDs enthält. Nur wenn diese ID im Server eingetragen ist, kann sich die Steuerung verbinden und es ist eine Verbindung von LASAL CLASS möglich.

1.10.1.2 lsrepeater.cfg

In dieser Datei müssen nachfolgende Schlüssel-Werte-Paare enthalten sein. Die obligatorischen Parameter müssen vorhanden sein, damit die Datei gültig ist und die Steuerung eine Verbindung zum Repeater-Server aufbaut und sich anmeldet.

Mit einer Raute „#“ können Kommentare gesetzt werden.

Obligatorische Parameter

```
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5510
REPEATER_ID = $SERIALNUM
PASSWORD = Sigmatek
```

Die REPEATER-IP enthält die IP und REPEATER-PORT den Port des Repeaters, der benutzt werden soll. Als IP kann auch eine DNS angegeben werden.

Unter REPEATER-ID kann eine individuelle ID vergeben werden. Diese ID muss numerisch sein mit maximal acht Stellen. Wird als Wert \$SERIALNUM angegeben, meldet sich die Steuerung beim Repeater mit der Seriennummer an.

PASSWORD ist das Passwort, welches auf dem Server für die ID gesetzt wurde. Diese wird beim Verbindungsauflauf abgefragt (hinterlegt auf dem Repater-Server). Das Passwort ist nicht das Online-Passwort welches beim Onlinegehen über LASAL CLASS eingetragen und in der Steuerung über „set onlinepwd“ gesetzt wird.

Optionale Parameter

```
KEEPALIVE_INTERVAL = 30      #Default 20
KEEPALIVE_RETRY = 30         #Default 20
KEEPALIVE_TMO = 30           #Default 20
TCP_WINDOW_SIZE = 12288     #Default 32k
```

KEEPALIVE_INTERVAL: Die Zeit in Sekunden, welche verstreichen muss, bevor ein Keep-Alive-Paket an den Repeater-Server gesendet wird.

KEEPALIVE_RETRY: Die Zeit in Sekunden zwischen 2 Keep-Alive-Paketen.

KEEPALIVE_TMO: Wenn die Gegenstelle nicht erreichbar ist, wird nach dem Verstreichen dieser Zeit (in Sekunden) die Verbindung geschlossen.

TCP_WINDOW_SIZE: Die Größe vom Window im TCP/IP Protokoll (kann zwischen 4*1460 und 40 kByte sein).

1.10.1.3 Kommando

```
lslrpt start
```

Mit lslrpt start CLI oder in der Autoexec.lsl wird die Funktion gestartet.

```
lslrpt stop
```

Mit lslrpt stop CLI oder in der Autoexec.lsl wird die Funktion deaktiviert.

Unter Umständen muss noch zusätzlich ein Gateway eingestellt werden (diese Information bitte beim Netzwerkadministrator erfragen).

```
set ip gateway 192.168.1.1
```

1.10.2 Beispiel-Konfigurationen mit der **lslrepeater.cfg**-Datei

Die lslrepeater.cfg Datei wird benötigt, um die Steuerung über einen Repeater zu erreichen (siehe Repeater).

1.10.2.1 Szenario 1: Steuerung meldet sich mit Seriennummer an

Im ersten Szenario meldet sich die Steuerung am Repeater mit der Seriennummer an. Die folgenden Zeilen müssen deshalb an die lslrepeater.cfg-Datei angefügt werden:

```
##### Configuration - Repeater Logged-On with Serial number#####
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5510
REPEATER_ID = $SERIALNUM
PASSWORD = Sigmatek
```

1.10.2.2 Szenario 2: Steuerung meldet sich mit individueller ID an

Im zweiten Szenario meldet sich die Steuerung am Repeater mit einer individuellen ID (113355) an.

Die folgenden Zeilen müssen deshalb an die lslrepeater.cfg-Datei angefügt werden:

```
##### Configuration - Repeater Logged-On with individual ID#####
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5510
REPEATER_ID = 113355
PASSWORD = Sigmatek
```

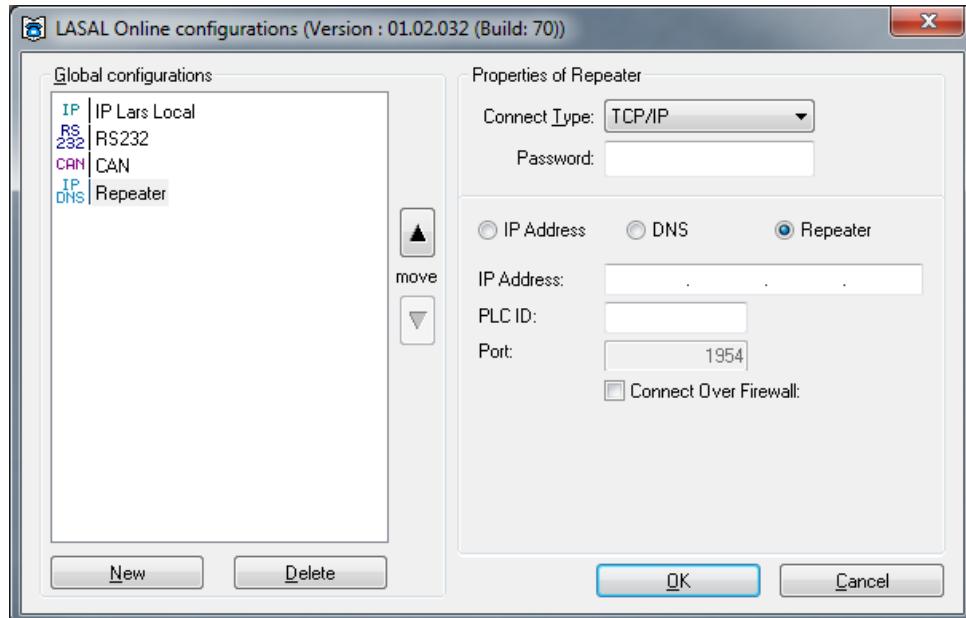
1.10.2.3 Szenario 3: Steuerung meldet sich über DNS an

Im dritten Szenario meldet sich die Steuerung am Repeater über einen Namen (DNS) an. Die folgenden Zeilen müssen deshalb an die `lslrepeater.cfg`-Datei angefügt werden.

```
##### Configuration - Repeater Logged-On with DNS#####
REPEATER_IP = Sigmatek.at/repeater
REPEATER_PORT = 5510
REPEATER_ID = $SERIALNUM
PASSWORD = Sigmatek
```

1.10.2.4 LASAL Tools

Im LASAL Online Configuration wurde die TCP/IP-Verbindung um den Repeater erweitert. Bei IP Adresse wird die Repeater-Server-IP eingetragen.



Im Feld PLC ID wird die Nummer, unter dem sich die Steuerung auf dem Repeater-Server angemeldet hat (entspricht der REPEATER_ID in der `lslrepeater.cfg` Datei) eingetragen.

Das Feld PASSWORD kann leer bleiben, wenn keine Online-Passwort vergeben wurde. Es ist somit abhängig vom Passwort, welches mit `set onlinepwd` in der Steuerung hinterlegt wurde. Dieses Passwort hat nichts mit dem Repeater und dem darin vergebenen Passwort zu tun (diese wird im `lslrepeater.cfg` definiert).

Auch das Feld Connect Over Firewall hat nicht direkt etwas mit der Repeater-Konfiguration zu tun und kann unverändert bleiben (es besteht die Möglichkeit, dass auf der eingetragenen Verbindung eine Firewall Pakete nicht weiterleitet - siehe Dokumentation LASAL CLASS).

1.10.3 LASAL Repeater

Der Server wurde in den bereits bestehenden VNC-Repeater integriert und eine Erweiterung vom Userinterface durchgeführt. Im Repeater müssen die IDs und dazugehörigen Passwörter der Steuerungen abgelegt sein. Für die Freigabe von IDs kontaktieren Sie bitte den Provider des jeweiligen Servers.

1.11 Remote Diagnose

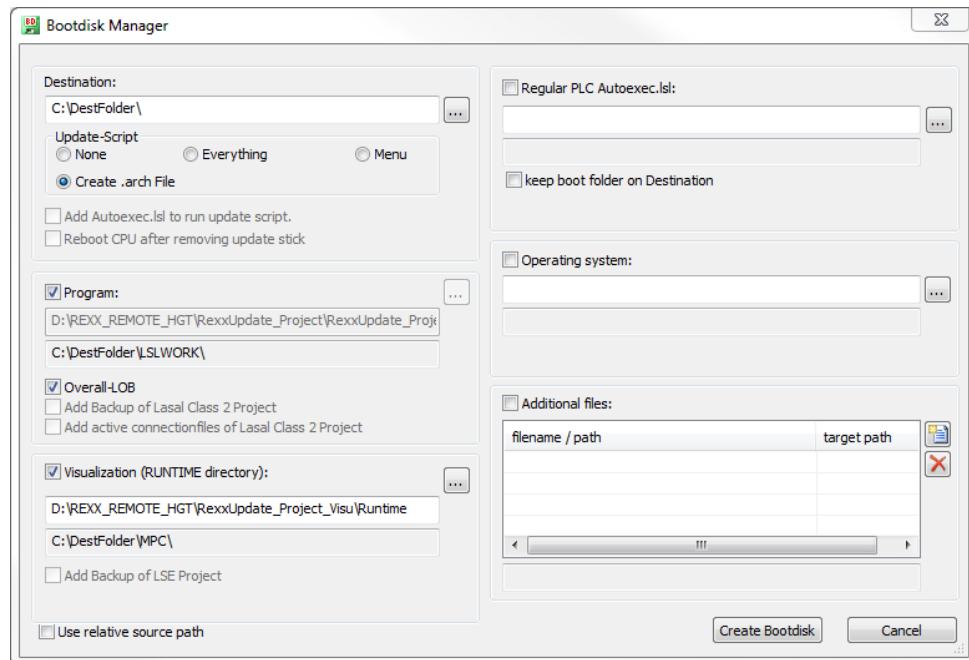
Bei der Remote-Diagnose wird eine Diagnosesteuerung über eine Ethernet-Verbindung an eine zu testende Steuerung angeschlossen. Nach dem Einschalten der Diagnosesteuerung wird ein Script ausgeführt, welches automatisch den Prüfling nach einer neuen Applikation überprüft (wird durch eine übertragenes Checksummandatei geprüft). Wird eine neue Applikation (als .arch-Datei verpackt) gefunden, wird diese Datei auf die Diagnosesteuerung übertragen, entpackt und gestartet.

Die „Diagnose-Applikation“ ist projektspezifisch zu erstellen. Es ist darauf zu achten, dass die Visualisierung als Mehr-CPU Lösung ausgeführt wird, damit die Diagnosesteuerung die Werte des Prüflings darstellen kann. Hierzu ist im LSE bei „Reference to Variables“ die IP-Adresse des Prüflings anzugeben.

Mögliche Fehlerquelle – im LSE-Projekt wird die IP-Adresse fix eingetragen, stellt man diese beim Kunden um, so passt unter Umständen die IP nicht mehr zur Anlage – keine Kommunikation!

Installing the Controls

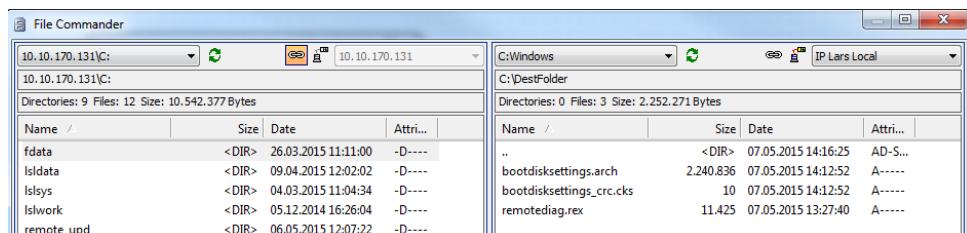
In LASAL CLASS 2 kann mit der Funktion CreateBootdisk (Menüleiste Tools => Create-Bootdisk) eine Diagnose-Applikation archiviert werden (.arch-Datei), gleichzeitig wird eine Checksummandatei (.cks-Datei) und ein Diagnose-Script (.rex-Datei) generiert. Die erstellten Dateien werden in den Ordner gelegt, der im Bootdisk Manager im Feld „Destination“ angegeben wird.



Im Bereich „Update-Script“ muss das Auswahlfeld „Create .arch File“ aktiviert werden um aus dem unter „Programm“ angegebenen Projektfolder (.lcp-File) und der (optional) ausgewählten Visualisierung (Runtime-Ordner) eine komprimierte Archiv-Datei (.arch) erstellen zu können. Außerdem wird eine Checksummdatei und ein Diagnose-Script in den angegebenen Zielordner kopiert.

Durch Drücken der Taste „Create Bootdisk“ werden die Dateien erstellt.

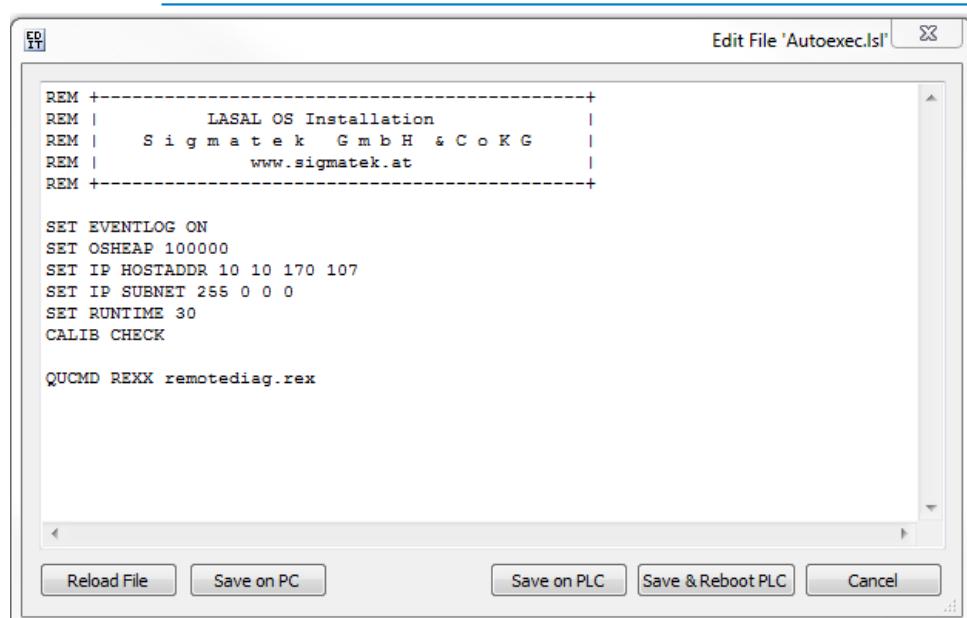
Wenn die Meldung „Create bootdisk finished successful!“ erscheint, können die erzeugten Dateien aus dem Zielordner entnommen und auf die entsprechenden Steuerungen kopiert werden.



1. Die Dateien „bootdisksettings.arch“ und „bootdisksettings_crc.cks“ müssen auf der zu testenden Steuerung (z.B.: CP 111) im Hauptverzeichnis „C:\“ in den Ordner „remote_upd“ gelegt werden (falls dieser Ordner nicht existiert, kann er mit der FileCommander-Funktion „MAKE DIR“ erstellt werden). Bei Bedarf kann der Speicherort und Name des Ordners auch in der .rex-Datei „remoteDiag.rex“ geändert werden (siehe Rexx-Konfigurationen).
2. Die Datei „remotediag.rex“ muss auf der Diagnosesteuerung (z.B.: HGT 835) in das Hauptverzeichnis „C:\“ kopiert werden.



In der Datei autoexec.lsl auf der Diagnosesteuerung muss außerdem der Aufruf „QUCMD REXX remotediag.rex“ vorhanden sein, damit der Diagnosevorgang automatisch nach dem Hochfahren gestartet wird!



```
REM +-----+
REM |           LASAL OS Installation           |
REM |           S i g m a t e k   G m b H   & C o K G   |
REM |           www.sigmatek.at                  |
REM +-----+
SET EVENTLOG ON
SET OSHEAP 100000
SET IP HOSTADDR 10 10 170 107
SET IP SUBNET 255 0 0 0
SET RUNTIME 30
CALIB CHECK

QUCMD REXX remotediag.rex
```

3. Rexx Konfigurationen: Am Beginn des Scripts befindet sich der Bereich für Anwender-Definitionen (diese Definitionen beziehen sich auf die zu testende Steuerung).

Hier muss die entsprechende IP-Adresse des Prüflings eingestellt werden.

Auch der Pfad, in dem die .arch-Datei und die Checksummandatei liegen, kann geändert werden.

```
/*-----User Definitions-----  
interface = 'IP' /*interface for online connection*/  
address = '10.10.170.200' /*address of interface for online connection*/  
srcFolder = 'c:\remote_upd' /*path for diagnostic-application*/  
/*-----
```

2 Betriebssystem-Schnittstellen

2.1 Variablen

Location	Datentyp	Beschreibung
_RealMaximumTime	(*AT % M 0000*)	UDINT (4 bytes unsigned)
_RealAverageTime	(*AT % M 0004*)	UDINT (4 bytes unsigned)
_CyclicMaximumTime	(*AT % M 0008*)	UDINT (4 bytes unsigned)
_CyclicAverageTime	(*AT % M 000C*)	UDINT (4 bytes unsigned)
_BackgroundMaximumTime	(*AT % M 0010*)	UDINT (4 bytes unsigned)
_BackgroundAverageTime	(*AT % M 0014*)	UDINT (4 bytes unsigned)

Diese Werte sind für alle SPS bis LasalOS Version 01.01.050 immer verfügbar.

Seit LasalOS Version 01.01.050 für kleine CPUs (CCL081, CCL721, CCL722, DCL641, DCC080-L) sind diese Tasks standardmäßig deaktiviert (aus Performance-Gründen) und muss über folgende Systemvariablen aktiviert werden:

_SysTaskMeasEnable	(*AT % M 0319*)	USINT (1 bytes unsigned)	Aktiviert / Deaktiviert die Messung
--------------------	-----------------	--------------------------	-------------------------------------

Ist diese auf 0 gesetzt, wird die Messung von maximaler und durchschnittlicher Zeit bei Realtime, Cyclic und Hintergrund deaktiviert (Leistungssteigerung bei kleineren CPUs), ansonsten werden diese aktiviert.

_rebootOnError	(*AT % M 0316*)	USINT (1 bytes unsigned)	Ermöglicht Neustart im Fehlerfall
----------------	-----------------	--------------------------	-----------------------------------

Der Standardwert ist 0, kein Neustart im Fehlerfall. Bei der Einstellung auf einen Wert $\neq 0$, startet das System nach einigen Sekunden neu.

_isRebootCodeAvailable	(*AT % M 0317*)	USINT (1 bytes unsigned)	
------------------------	-----------------	--------------------------	--

Ist diese Variable 1, wird die Variable _rebootCode gültig.

_rebootCode	(*AT % M 0317*)	USINT (1 bytes unsigned)	Aktiviert / Deaktiviert die Messung
-------------	-----------------	--------------------------	-------------------------------------

Diese Variable enthält den CPU-Status vor dem Neustart, falls ein Fehler aufgetreten ist. Mit diesem Wert kann man den Grund des Neustarts erfahren.

_RtOSversion	(*AT % M 02EC*)	UDINT (4 bytes unsigned)	Version des SPS-Betriebssystems.
--------------	-----------------	--------------------------	----------------------------------

Einstellung der _RtOSversion Variable (4-Byte):

```
3 2 1 0 nth byte der Variable
+---+---+---+---+
| 0 | 0 | 5 | 11 |
+---+---+---+---+
|   |   |   +-----> Minor Nummer des OS
|   |   +-----> Major Nummer des OS
|   +-----> Nicht verwendet
+-----> Nicht verwendet
```

_UserProgPointer	(*AT % M 02F0*)	^USINT (Zeiger auf 4 bytes unsigned)	Adresse des Benutzer Codes.
_UserProgSize	(*AT % M 02F4*)	UDINT (4 bytes unsigned)	Größe des User-ausführbares Programms (Code).
_UserDataPointer	(*AT % M 02F8*)	^USINT (Zeiger auf 4 bytes unsigned)	Adresse des User-Datenspeicher.
_UserDataSize	(*AT % M 02FC*)	UDINT (4 bytes unsigned)	Größe des User-Datenspeicher.
_cpuLoad	(*AT % M 0EA6*)	UINT (2 byte unsigned);	CPU load.

Diese Variable enthält die CPU-Auslastung in Promille (0-1000). Dieser Wert wird circa alle 250 ms aktualisiert.

_bruntime	(*AT % M 030A*)	UDINT (4 byte unsigned);	Hintergrund Runtime-Zähler
-----------	-----------------	--------------------------	----------------------------

Diese Variable enthält den Hintergrund Runtime-Zähler. Ab dem programmierbaren Wert wird der Zähler in 10 ms Schritten dekrementiert. Ist der Wert 0, löst die CPU einen Hintergrund Runtime Fehler (Error 68) aus. Der Zähler wird mit dem programmierbaren Wert bei jedem Programmzyklus neu geladen.

_swbruntime	(*AT % M 030E*)	UDINT (4 byte unsigned);	Sollwert für den Hintergrund Runtime-Zähler
-------------	-----------------	--------------------------	---------------------------------------------

Diese Variable enthält den programmierbaren Wert für den Hintergrund Runtime-Zähler. Die Variable kann während der Laufzeit verändert werden. Setzen Sie diese Variable auf

"0", um den Hintergrund Runtime-Zähler zu deaktivieren. Die Standardeinstellung ist Off (0).

_runstatus	(*AT % M 0EAA*)	USINT (1 byte unsigned);	Aktueller Run-Status
------------	-----------------	--------------------------	----------------------

Diese Variable enthält den aktuellen Fehlerwert der CPU in der Statusanzeige bzw. in der LASAL Software in der Statuszeile.

0	Programm wurde aus dem RAM gestartet.
1	Programm wurde vom ROM gestartet.
2	Runtime Error
3	Pointer
4	Checksummen Fehler
usw.	

_ClockTicks	(*AT % M 0EAB*)	UINT (2 byte unsigned);	Takt in μ s
-------------	-----------------	-------------------------	-----------------

Diese Variable enthält den Taktzyklus in μ s.

_runtime	(*AT % M 0EAE*)	USINT (1 byte unsigned);	Runtime Zähler.
----------	-----------------	--------------------------	-----------------

Diese Variable enthält den Runtime-Zähler. Ab dem programmierbaren Wert wird der Zähler in 10 ms Schritten dekrementiert. Ist der Wert 0, löst die CPU einen Runtime-Fehler (Fehler 02) aus. Der Zähler wird mit dem programmierbaren Wert bei jedem Programmzyklus neu geladen.

_swruntime	(*AT % M 0EAF*)	USINT (1 byte unsigned);	Sollwert für den Runtime-Fehler Zähler.
------------	-----------------	--------------------------	-----------------------------------------

Diese Variable enthält den programmierbaren Wert für den Runtime-Zähler. Die Standardeinstellung ist 255.

_error_cnt_dias	(*AT % M 0E94*)	USINT (1 byte unsigned);	Error Zähler für DIAS-Bus.
-----------------	-----------------	--------------------------	----------------------------

Diese Variable enthält die aktuelle Position des Fehlerzählers für den DIAS-Bus. Jeder Re-entry Versuch am DIAS-Bus erhöht den Wert. Der Zähler ist als 8-Bit Endloszähler konstruiert.

_FirstScan	(*AT % M 0EB8*)	USINT (1 byte unsigned);	Hoch bei der Programminitialisierung.
------------	-----------------	--------------------------	---------------------------------------

Der erste Scan ist eine Byte-Adresse im Datenspeicher, der nur für die Dauer des ersten Programmzyklus nach dem Start des Programms durch das Betriebssystem auf 1 gesetzt wurde.

_IOsegment	(*AT % M 0EBA*)	UDINT (4 bytes unsigned);	IO-Segment Adresse.
------------	-----------------	---------------------------	---------------------

Diese Variable enthält die Segment-Adresse der DIAS-Peripherie. Dies ist erforderlich, um die Ausführung der direkten Adresszugriffe auf das DIAS-Funktionsmodul durchzuführen.

_DIASconfig	(*AT % M 0EC0*)	ARRAY [0-63] OF USINT (64 1-byte fields unsigned)	Enthält DIAS-Hardware-IDs für jede mögliche DIAS- Stationsnummer.
-------------	-----------------	------------------------------------------------------	-------------------------------------------------------------------------

Dieses Array enthält die Systemkonfiguration ihres DIAS-Systems. Jedes Byte stellt im System eine Rufnummer (0 Byte ... Station 00, 63 Byte ... Station 63) dar. Der Wert in einem Byte gibt Auskunft über die auf dieser Station befindlichen DIAS-Karten.

_CDIASconfig	(*AT % M 0f00*)	: ARRAY [0-7] OF USINT (1-byte fields unsigned)	Enthält C-DIAS Hardware- IDs für jede mögliche DIAS-Stationsnummer.
--------------	-----------------	----------------------------------------------------	---------------------------------------------------------------------------

Dieses Array enthält die Systemkonfiguration ihres C-DIAS-Systems. Jedes Byte stellt im System eine Rufnummer (0 Byte ... Station 00, 7 Byte ... Station 7) dar.

Der Wert in einem Byte gibt Auskunft über die C-DIAS-Karte die auf dieser Station ist. Diese Funktion ist ab LasalOS Version 01.01.056 verfügbar.

_CpuDisplay	(*AT % D 0001*)	USINT (1 byte unsigned)	Display auf x 386 PLCs.
-------------	-----------------	-------------------------	-------------------------

Die DISPLAY-Variable stellt die D 0001 Adresse dar.

Der Inhalt dieser Adresse wird in der CPU-Anzeige als Dezimalwert dargestellt. Allerdings können nur die Werte 00-99 angezeigt werden. Das Display bleibt dunkel bei Werten über 99.

OPS	(*AT % D 00003*)	OPSys (pointer on struct OPSys)	Real-time Task Daten.
-----	------------------	------------------------------------	-----------------------

2.1.1 Inhalt der OP-Struktur

tAbsolute: UDINT (4-Byte-unsigned) Absolute Zeit vom Start in ms.
SysState: CONFSTATES LASAL Runtime-Status.

ConfigTime: UDINT; Konfigurationszeit.

OverRun: UINT Zählt Überschreitungen.

RtInterv_mSec: UDINT Immer auf 1 gesetzt.
RtInterv_uSec: UDINT Immer auf 1000 gesetzt.

udDescCRC: UDINT CRC des laufenden Projekts, dieser wird
aus allen Schlagwortlisten berechnet.

```
uiLoaderVersion : STRUCT    Loader-Version Eintrag.
usLoRev: USINT;
usHiRev: USINT;
END_STRUCT;
END_STRUCT;
```

_Isl_pOS	(*AT % M 0EA2*)	\LSL_OSDATA (Zeiger auf die OS interface structure)	_Isl_pOS
----------	-----------------	-----------------------------------------------------	----------

Das ist die LASAL API, über die die Bibliothek auf die RTOS LASAL OS-Funktionen zugreift. Siehe RTOS-lib Dokumentation für weitere Informationen.

_UserHeapStartAddr	(*AT % M 0020*)	UDINT (4 bytes unsigned)	Start-Adresse der UserHeapMemory
_UserHeapTotalSize	(*AT % M 0024*)	UDINT (4 bytes unsigned)	Gesamtgröße der UserHeapMemory
_heapAllocCnt	(*AT % M 0312*)	UDINT (4 bytes unsigned)	Anzahl der UserHeap Allokationen
_UserHeapUsedMem	(*AT % M 0028 *)	UDINT (4 bytes unsigned)	Anzahl des genutzten Speichers von UserHeap

Diese Variable enthält die Größe des verwendeten Speichers und Informationen der internen Speicherverwaltung. Aus diesem Grund wird über 100 Byte Speicher benötigt, wenn z.B. 100 Bytes vom User-Heap zugeteilt werden.

_UserHeapFreeMem	(*AT % M 002C*)	UDINT (4 bytes unsigned)	Menge von freiem Speicher im UserHeap
------------------	-----------------	--------------------------	---------------------------------------

Diese Variable enthält die Summe aller freien Speicherblöcke. Wenn z.B. 100 Bytes des Speichers frei sind, ist die zugewiesene Speichergröße kleiner als dieser Wert.

_MaxDataMem	(*AT % M 003C*)	UDINT (4 bytes unsigned)	Maximaler Datenspeicher der Projekte
-------------	-----------------	--------------------------	--------------------------------------

Diese Variable enthält den maximalen verwendeten Speicher für die System- und Globalvariablen.

_MaxCodeMem	(*AT % M 0040 *)	UDINT (4 bytes unsigned)	Maximaler Code-Speicher der Projekte
-------------	------------------	--------------------------	--------------------------------------

Diese Variable enthält den maximalen Code-Speicher für die Anwendung.

_WhoAmI	(*AT % M 02E8*)	UDINT (4 bytes unsigned)	Informationen über den CPU-Typ
---------	-----------------	--------------------------	--------------------------------

Diese Variable enthält den CPU-Typ in Bezug auf einen Ganzzahlwert. Die Definition und Zuordnung der CPU-Type sind in der Datei Isl_st_kernel.h zu finden.

_S_RAM_Hptr	(*AT % M 0300 *)	^MRAM_DESCR (Zeiger auf MRAM_DESCR Struktur)	Zeiger auf S_RAM_Header
-------------	------------------	----------------------------------------------	-------------------------

Diese Struktur enthält Informationen über die SRAM-Daten wie Start-Adresse, Datenlänge und die Menge der verwendeten Daten.

_RTrlntVal	(*AT % M 0306 *)	UDINT (4 bytes unsigned)	Variable nicht verwendet
_OnlineMap	(*AT % M 0E96 *)	^_OnlineMap (Zeiger auf _OnlineMap Struktur)	Zähler von Online Benutzern

2.1.2 Inhalt der _OnlineMap-Struktur

bySerial	USINT	Serieller Online-Zähler
byTCP	USINT	TCP Online-Zähler
byCAN	USINT	CAN Online-Zähler
byReserve	USINT	nicht verwendet

2.2 API MultiTask

2.2.1 Allgemeine Hinweise

2.2.1.1 Multitasking

Was ist Multitasking?

Ein Thread, auch Task genannt, ist ein sequentieller Pfad zur Ausführung von Anweisungen. Die diskreten Statements eines Threads werden sequentiell nach der syntaktischen Struktur von C, C++ oder Pascal verarbeitet. Der Begriff Multitasking bedeutet, dass mehrere sequentielle Aufgaben parallel verarbeitet werden. Allerdings können in Einzelprozessor-Systemen mehrere Tasks nicht gleichzeitig ausgeführt werden. Daher müssen Task-Wechsel ausgeführt werden. Dies ist die Aufgabe eines Multitasking-Systems wie LASAL-OS. In vielen Fällen können Tasks nicht völlig unabhängig voneinander ausgeführt werden, aber es wird trotzdem erwartet, dass sie zusammenarbeiten. Beispielsweise kann es verlangt werden, dass ein bestimmter Task weitergeführt wird, nachdem ein anderer Task eine bestimmte Operation abgeschlossen hat. In einem solchen Fall müssen die betroffenen Tasks synchronisiert werden. Z. B.: Wird Parallelität von Tasks beschränkt, kann die Synchronisation mit Inter-Task

Kommunikation erfolgen. Für ein klares Verständnis von paralleler Programmierung ist es wichtig, dass man sich über die unterschiedlichen Anforderungen von Multitasking-Systemen bewusst ist. Die beiden folgenden Abschnitte beschreiben die wichtigsten Unterschiede zwischen Timesharing- und Echtzeitsysteme.

Timesharing

Timesharing nutzt Multitasking um das gleichzeitige Teilen eines leistungsstarken Computers von mehreren Benutzern (oder Batch-Jobs) zu ermöglichen. In der Regel führen Tasks unabhängig voneinander Timesharing-Systeme. Deshalb ist Inter-Task Kommunikation nur in einer einfachen Konfiguration verfügbar. Eine der beliebtesten Timesharing-Systeme ist Unix, das entwickelt wurde, als die Rechenleistung von verfügbaren Computern kleiner als die Anforderungen des Jobs wurde. Als Folge musste das Multitasking-System die begrenzte Rechenleistung so fair wie möglich unter den konkurrierenden Tasks teilen. Anzumerken ist, dass der Systemablauf normalerweise durch Multitasking in solchen Systemen degradiert wird, da der Zuordnungsaufwand erheblich ist. Unter Unix ist es z. B. möglich, dass zwei computergebundene Programme jedes davon eine Minute lang läuft, wenn es das einzige aktive Programm ist. Laufen diese parallel würden sie 2,5 Minuten brauchen. Parallel Verarbeitung kann daher fairer sein, da User 1 und User 2 gleich lang warten müssen, damit die Aufgaben abgeschlossen werden. Diese Art von Verarbeitung ist aber normalerweise weniger effizient.

Real-Time Systeme

Echtzeitsysteme erfüllen ganz andere Anforderungen. Ein Echtzeitsystem darf nie überlastet werden. Sobald also keine Rechenleistung mehr zur Verfügung steht, wird die Echtzeit nicht mehr unterstützt. Zum Beispiel könnte eine Situation mit Echtzeitanforderungen wie folgt auftreten: Ein Zähler generiert jede Sekunde Daten, die der Computer dann sammeln, verarbeiten und speichern muss. Wenn die Verarbeitung eines Datensatzes mehr als eine Sekunde auf dem Ziel-Computer erfordert, ist das System überlastet und die Echtzeit kann nicht mehr unterstützt werden. Echtzeitsysteme kümmern sich nicht um "Fairness". Tasks können Prioritäten haben, die streng befolgt werden müssen. Ein Task mit hoher Priorität kann die CPU-Zeit eines anderen Tasks mit einer niedrigeren Priorität jeder Zeit nehmen, ohne den anderen Task "fair" zu behandeln. Da Überlastung nicht erlaubt ist, werden Tasks mit niedriger Priorität eventuell weniger CPU-Zeit erhalten. Eine zusätzliche Eigenschaft von Echtzeitsystemen ist, dass sie innerhalb einer vorgegebenen Zeitspanne, die normalerweise kurz ist, auf Ereignisse reagieren müssen (bzw. erkennen, wenn das nicht eingehalten wurde). Externe Events werden, wenn möglich, mittels Interrupts verarbeitet. Daher haben Interrupt-Handler die strengsten Echtzeitanforderungen. Aus diesem Grund müssen Echtzeitsysteme eine

geringe Interrupt-Latenzzeit haben (d.h. die Zeit zwischen dem Interrupt-Signal und der Ausführung des ersten Interrupt-Handler). Für das Verhältnis der Tasks ist es wichtig, zwischen kooperativem und präemptivem Scheduling zu unterscheiden. Im präemptivem Scheduling ist die Reaktionszeit des Tasks auf Interrupts gleich der Interrupt-Latenz und der Umschaltzeit des Tasks. Im kooperativen Scheduling wird die maximale Zeitspanne zwischen zwei Kernelanrufe addiert.

Kooperativen und präemptives Multitasking

Präemptives Multitasking bedeutet, dass die Task-Wechsel vom Interrupt-Handler direkt ausgelöst werden können. Mit kooperativem (nicht präemptivem) Multitasking, wird ein Task-Wechsel nur durchgeführt, wenn ein Task den Kernel auruft. Das heißt, es verhält sich "kooperativ" und gibt dem Kernel eine Chance auf einen Task zu wechseln. Beispiel: Ein Receive-Interrupt-Handler für einen seriellen Port schreibt Daten auf eine Mailbox. Wenn ein Task auf die Mailbox wartet, wird er beim präemptiven Scheduling sofort vom Scheduler aktiviert. Beim kooperativen Scheduling, wird der Task nur auf den "Ready" Status gesetzt. Ein Taskwechsel wird nicht sofort durchgeführt. Nachdem der Interrupt-Handler abgeschlossen ist, läuft der unterbrochene Task weiter. Diese Art von "pending" Task-Wechsel wird später durch den Kernel ausgeführt, sobald es vom aktiven Task aufgerufen wird. LASAL-OS unterstützt sowohl kooperatives als auch präemptives Scheduling. Die Multitasking-Funktion ist standardmäßig auf kooperative Planung eingestellt.

Real-Time

Der Begriff "real time" wird häufig verwendet, aber selten definiert. Eine mögliche Definition ist:

Real-time Software kann mit Events außerhalb des Prozessors synchron laufen. Die maximale Reaktionszeit für externe Events ist vorhersehbar. Diese Definition sagt jedoch nichts über Multitasking aus. Multitasking ist nicht unbedingt erforderlich um real-time Software zu entwickeln, allerdings kann Multitasking die Entwicklung von real-time Software vereinfachen. Multitasking kann exzellente Reaktionszeiten erreichen, auch wenn andere Jobs parallel zu der real-time Verarbeitung durchgeführt werden müssen. Die Anforderungen für real-time Verarbeitung sind eine ausreichend kleine Interrupt-Latenz, eine konstante Task-Wechsel Zeit, die so kurz wie möglich sein sollte und (in vielen Fällen) präemptives Scheduling. Aus diesem Grund können komplexe Operationen mit nicht-deterministischen Laufzeitanforderungen, wie das Laden eines Prozesses von der Festplatte, nicht von einem real-time System während der Ausführung eines Task-Wechsels unterstützt werden.

LasalOS-Scheduler

Der Scheduler entscheidet, welcher Task ausgeführt wird. In diesem Zusammenhang werden nur Task-Zustände und Prioritäten berücksichtigt. Im Gegensatz zu vielen anderen Systemen ist LasalOS nicht vornehmlich auf Timer-Interrupt basiert. Stattdessen ist LasalOS ein Ereignis-gesteuertes System. Ein Ereignis ist eine Inter-Task Kommunikation, die durch einen Task oder ein Interrupt-Handler eingeleitet werden kann und zu Zustandsänderungen der beteiligten Tasks führen kann. Der LasalOS Timer Interrupt-Handler ist nur ein Interrupt-Handler unter vielen. Sein Zweck ist es, Tasks zu einer bestimmten Zeit auf den "Ready" Status zu stellen. Einige Multitasking-Programme können komplett ohne den Timer Interrupt-Handler laufen.

Der Scheduler hält sich an die folgenden Regeln:

1. Der Task mit der höchsten Priorität wird vor allen anderen ausgeführt.
2. Wenn der Scheduler die Wahl zwischen mehreren Ready-Tasks mit der gleichen Priorität hat, wird der am längsten wartende Task aktiviert.
3. Wenn mehrere Tasks auf ein Ereignis warten, werden sie bei der Auslösung des jeweiligen Events entsprechend ihrer Priorität aktiviert.
4. Mit Ausnahme des Time-Slice Taskswitches werden Task-Wechsel nur ausgeführt, wenn Regel 1 verletzt werden würde. Die Zahl der Task-Wechsel wird somit minimiert.

Normalerweise werden die oben genannten Regeln ohne Ausnahme befolgt. Wann immer ein Task-Zustand sich ändert, prüft der Scheduler ob ein Task-Wechsel nach den Scheduling Regeln notwendig ist. Nur beim kooperativen Scheduling kann Regel 1 zwischen einem Interrupt und des nächsten Aufrufs an den Kernel verletzt werden.

Task-Wechsel

Es gibt drei wichtige Arten von Task-Wechsel im LasalOS: Blockierung, Aktivierung und Time Slice. Ein Task wird immer an dem Punkt fortgesetzt, wo er vor einem Task-Wechsel gestoppt wurde. Unter den folgenden Bedingungen treten drei Arten von Task-Wechsel auf:

Blockierender Task-Wechsel	Ein blockierender Task-Wechsel wird ausgelöst, wenn ein Task selbst blockiert und nicht weitergeführt werden kann. Dies kann, zum Beispiel durch den Versuch aus einer leeren Mailbox Daten abzurufen erfolgen oder wenn ein Task CPU-Zeit für eine gewisse Zeit durch den Aufruf der Funktionen von TASKDELAY frei gibt. In diesem Fall wird LasalOS den Task mit der höchsten Priorität ausführen der sich im Ready Zustand befindet (Scheduling Rules). Falls keiner der Tasks bereit ist, wird der Idle Task aktiviert.
Aktivierender Task-Wechsel	Task-Wechsel werden ausgelöst, wenn ein Task mit einer höheren Priorität als der aktuelle Task im Ready-Zustand ist. Ein Task-Wechsel wird zum Beispiel ausgelöst, wenn ein Task mit einer

**Time Slice
Task-Wechsel**

hohen Priorität auf Daten aus einer Mailbox wartet und ein anderer dort Daten hinterlegt hat. Der wartende Task kann jetzt mit dem Abrufen der Daten fortfahren und wird sofort aktiviert.

Der LasalOS Timer Interrupt-Handler führt Time Slice Task-Wechsel aus, wenn die oben stehenden Bedingungen erfüllt sind. Kooperative Time-Slice Task-Wechsel können auch direkt durch den Aufruf des TASKDELAY (0) ausgelöst werden.

Neben den oben genannten Arten von Task-Wechsel wird in präemptive sowie kooperative Task-Wechsel unterschieden. Präemptive Task-Wechsel werden durch einen Interrupt-Handler ausgelöst, während kooperative Task-Wechsel von Tasks initiiert werden. Blockierende Task-Wechsel sind immer kooperativ, da sie nicht von Interrupt-Handler initiiert werden dürfen. LasalOS betrachtet einen präemptiven, blockierenden Task-Wechsel als Fehler. Die Aktivierung von Tasks mittels dieser Mechanismen kann in beiden Formen auftreten. Im ersten Fall ist der Task-Wechsel kooperativ; im letzteren ist er präemptiv.

LasalOS kann für kooperatives- oder präemptives Scheduling konfiguriert werden. Wenn präemptives Scheduling deaktiviert ist, werden präemptive Task-Wechsel vom Kernel verzögert, bis der aktive Task einen Scheduler-Funktion (ein LasalOS-API-Funktion) aufruft. Potentielle präemptive Task-Wechsel werden daher in kooperative Task-Wechsel umgewandelt, wenn die Applikation präemptives Scheduling nicht aktiviert hat.

Time Slice Task-Wechsel werden durchgeführt, wenn die folgenden Bedingungen erfüllt sind:

1. Time-Slicing muss eingeschaltet werden (die Standardeinstellung ist ausgeschaltet)
2. Mindestens ein Task mit der gleichen Priorität wie der aktive Task muss im Ready-Zustand sein.
3. Der letzte Task-Wechsel muss an einem vom Time-Slice angegebenen Zeitpunkt aufgetreten sein. Der letzte Task-Wechsel wurde aufgrund eines anderen Events als Time-Slicing ausgelöst.

LasalOS setzt den Time-Slice Wert. Bevorzugungen sind für Time-Slicing nicht erforderlich. Time Slicing spielt nur eine kleine Rolle und darf aus diesem Grund nicht gegen die oben stehenden Regeln zum Scheduling verstößen.

LasalOS Tasks/Threads

Ein Task oder Thread ist eine C- oder Structured-Text-Funktion, die ihren eigenen Stack besitzt. Task Prioritätsstufen (für die Applikation) reichen von 1 bis 14, eine hohe Priorität bedeutet eine hohe Dringlichkeit für die Ausführung. Prioritäten sind nur in der Relation zueinander sinnvoll. Die Tasks werden referenziert mittels Task-Handles. Task-Handles sind vergleichbar mit File-Handles. Wenn ein Task erstellt wird, liefert LasalOS einen einzigartigen Handle, der später als Referenz verwendet werden kann (z.B. Daten

schicken). Mit der Ausnahme statischer Variablen werden alle lokalen Task-Variablen auf dem Stack angelegt. Dasselbe gilt für die lokalen Variablen aller Funktionen, die durch den Task aufgerufen werden. Mehrere Tasks können daher mit der gleichen Task-(Einsprungs-)Funktion gestartet werden. Jedem Task wird ein eigener Stack sowie eigenen lokalen Variablen zugeordnet. Eine einzige Funktion kann mittels unterschiedlichen Tasks aufgerufen werden, indem derselbe Code verwendet wird. Da jeder Task seinen eigenen Stack besitzt, treten keine Wiedereintrittsprobleme auf, solange die Funktion auf die eigenen Parameter und lokalen (nicht-statischen) Variablen zugreift. Die Sichtbarkeitsvorschriften von C/Structured-Text beziehen sich völlig auf Multitasking-Programme. Alle Tasks können auf globale Daten zugreifen. Wenn die Applikation initialisiert wird, werden drei Threads erstellt: Die RTWORK- und CYCLE- und BACKGROUND-Task. Dort werden die LASAL-Tasks abgearbeitet.

Ein Task hat immer einen der folgenden Status:

Strom	Der aktive Task ist im Current-Status und wird ausgeführt. Im LasalOS darf immer nur ein Task in diesem Zustand sein.
Ready	Alle Tasks, die zum Starten bereit sind, werden auf den Ready-Status gestellt. Normalerweise haben alle Ready-Tasks entweder die gleiche oder eine geringere Priorität als der aktive Task.
Suspended:	Suspendierte Tasks können nicht ausgeführt werden, weil sie durch LasalOS mit der SUSPEND-Funktion gestoppt wurden. Um suspendierte Tasks wieder ausführen zu können, muss die LASAL OS Resume-Funktion aufgerufen werden.
Blocked	Tasks, die blockiert sind, können nicht ausgeführt werden, weil sie auf ein Ereignis warten (z.B. ein Semaphor Signal oder eine hereinkommende Nachricht in die Mailbox). Diese Tasks können nur von einem anderen Task oder einem Interrupt-Handler zurückgesetzt werden.
Delaying	Diese Tasks haben sich selbst für eine bestimmte Zeitspanne blockiert. Sie werden automatisch vom LasalOS Timer-Interrupt-Handler zurückgesetzt, nachdem die Verzögerungszeit abgelaufen ist.
Timed	Timed-Tasks warten auf ein Ereignis oder ein Timeout. Solche Tasks werden auf Ready zurückgesetzt, wenn das Ereignis eintritt oder der Timeout aufgetreten ist.

LasalOS behält alle inaktiven Tasks in mehreren Warteschlangen. Zum Beispiel gibt es einen Task-Queue für alle Ready-Tasks und eine andere Warteschlange enthält alle Tasks, die für eine bestimmte Zeit warten. Warteschlangen werden auch bei Semaphoren oder Mailboxen gebildet, wenn Tasks dort blockieren.

LASAL Tasks

LASAL Tasks sind Methoden eines LASAL-Objekts (RtWork, CyWork oder Background), die vom Betriebssystem zyklisch aufgerufen werden. Wird die Anwendung initialisiert, schreibt der Loader den LASAL Task in die Task-Liste. Task-Listen sind für Real-Time, Cyclic- und Background Tasks verfügbar. Jede Task-List Eintrag enthält einen

Zeitabstand (Periode), der in LASAL Klasse (bzw. den Objekten) konfiguriert werden kann.

Die Applikations-Tasks werden in einer Endlosschleife ausgeführt, um jeden Eintrag (Objekt) zu prüfen, ob der Zeitabstand seit dem letzten Aufruf abgelaufen ist. Ist die angegebene Zeit abgelaufen, wird die Methode aufgerufen und der Zeitpunkt des letzten Aufrufs aktualisiert.

Nachdem die Task-Liste abgeschlossen ist (Applikation beendet/im Reset/im Error), werden die Applikationstasks für eine bestimmte Zeit weiterlaufen gelassen, damit die Task-Liste abgearbeitet werden kann. Der Real-Time-Task wird in einem Zeitraum von 1 ms in IPCs und 2 ms für 386 CPUs ausgeführt. Bei aktuellen Plattformen wird dieser Wert konfiguriert (Tick: 125 µs - 2 ms). Der Zeitraum für den Cyclic-Task liegt bei mindestens 1 ms, kann aber höher liegen, wenn der Tick höher ist. Der Background-Task liegt in der Regel bei 2 ms.

Inter-Task Kommunikation

Der Begriff Inter-Task Kommunikation umfasst alle Komponenten, die für den Informationsaustausch zwischen den Tasks verwendet werden. LasalOS bietet drei verschiedene Techniken: Semaphoren, Mailboxes und Message-Passing.

Praktisch alle Multitasking-Systeme bieten Semaphoren, die für den Signalaustausch verwendet werden, um Tasks zu aktivieren oder zu blockieren. Ein Semaphor ist eine Variable in der ein Signalwert gespeichert oder gelesen werden kann. Task-Wechsel können auftreten, wenn auf ein Semaphor mit einem Wert von 0 zugegriffen wird. Es gibt fünf verschiedene Semaphoren-Typen im LASAL Betriebssystem: Zähler, Binär, Ereignis, Ressourcen und Mutex.

Mailboxen sind eine Erweiterung des Semaphoren-Konzepts. Anstelle von Signalwerten können Daten gespeichert oder aus einer Mailbox gelesen werden. Ein Task-Wechsel tritt auf, wenn auf eine leere oder volle Mailbox zugegriffen wird. Die Anzahl der Datensätze in einer Mailbox kann konfiguriert werden. Mailboxen sind besonders zur Pufferung von Daten zwischen Tasks oder Interrupt-Handler und Tasks nützlich.

Message Passing wird zum Datenaustausch unmittelbar zwischen zwei Tasks verwendet; keine Daten oder Signale werden gepuffert. Da die Tasks sich synchronisieren müssen, ist dies die engste Kopplung zwischen Tasks.

Reentrance

Der Begriff reentrance bezieht sich auf Probleme, die auftreten können, wenn mehrere Tasks den gleichen Code ausführen oder auf globale Daten gleichzeitig zugreifen. Reentrant Code bedeutet, dass ein Task den Code abarbeiten kann, bevor ein anderer

Task mit der Ausführung fertig ist. In einer Multitasking-Umgebung sollte darauf geachtet werden, dass so viel Code wie möglich reentrant ist, so dass er von mehreren Tasks verwendet werden kann.

Das folgende Beispiel verdeutlicht die Probleme, die auftreten können, wenn globale Daten verwendet werden. Vorausgesetzt wird, dass zwei Tasks zum Zählen verwendet werden und das Programm eine globale "Counter" Variable vom Typ int enthält, die mit einem Wert von 0 initialisiert wird. Beide Tasks werden die folgenden Anweisungen ausführen.

```
Counter = Counter + 1;
```

Immer wenn ein Ereignis eintritt, das gezählt wird.

Dieser Befehl könnte vom Compiler wie folgt übersetzt werden.

```
MOV EAX, Counter;      line 1  
ADD EAX, 1;           line 2  
MOV Counter, EAX;     line 3
```

Zur Vereinfachung wird angenommen, dass beide Tasks die gleiche Priorität haben, präemptives Scheduling aktiviert und das Time-Slicing aktiv ist.

Folgendes könnte auftreten: Task 1 erkennt ein Ereignis und beginnt die Ausführung der oben stehenden Maschinensprachanweisungen. Nachdem Zeile 1 ausgeführt wurde (Register EAX enthält 0), tritt ein Time-Slice-Task-Wechsel auf und Task 2 wird aktiviert. Aufgabe 2 erkennt auch einen Event und erhöht die Counter-Variable um 1, ohne unterbrochen zu werden. Später erhöht wieder Task 1 (in Zeile 2), EAX von 0 auf 1 und speichert diesen Wert in die Variable Counter.

Auch wenn Counter zweimal erhöht wurde hat er nur noch einen Wert von 1.

Dieses Beispiel zeigt, dass zwei oder mehr Tasks niemals auf globale Daten gleichzeitig zugreifen dürfen, wenn mindestens einer der Tasks die Daten ändern kann. In Wirklichkeit ist es nicht der Code, der nicht reentrant ist, sondern die vom Code manipulierten Daten. Auch wenn das Statement "Counter = Counter + 1;" in beiden Tasks getrennt ist, wird das Problem nicht gelöst. Infolgedessen sollten globale Daten nicht geteilt werden. Das Gleiche gilt für lokale Variablen, die als statisch deklariert sind. Wenn das Teilen von globalen Daten nicht vermieden werden kann, sollte es durch die Verwendung von Semaphoren geschützt werden. Leider gibt es einige Teile des Codes, die der Programmierer nicht kontrollieren kann. Zum Beispiel Run-Time Systembibliotheken.

2.2.1.2 Zeit

Für Real-Time Systeme ist die Zeit von großer Bedeutung. LASAL OS besitzt eine Interrupt-gesteuerte Uhr die eingestellt sowie gelesen werden kann. Darüber hinaus kann ein Task sich für einen bestimmten Zeitraum blockieren, um CPU-Zeit für andere Tasks freizugeben. Die Zeit ist in Timerticks angegeben. In LASAL CLASS wird die Zeit in

Millisekunden angegeben. LASAL OS empfängt seine Timer-Interrupts vom Clock-Gerätetreiber. Dieser wird nur benutzt, um periodische Interrupts zu erzeugen. LASAL OS weiß nicht wie viel Zeit (z.B. in Sekunden) tatsächlich zwischen zwei aufeinander folgenden Timer-Ticks vergeht.

2.2.1.3 Message Passing

Zusätzlich zu den Mailboxen bietet LasalOS mit Message Passing einen weiteren Mechanismus zur Inter-Task-Kommunikation. Mit Message Passing sind Datenobjekte wie Semaphoren oder Mailboxes zur zwischenzeitlichen Datenspeicherung nicht mehr erforderlich. Daten werden unmittelbar zwischen den Tasks kopiert. Message Passing ist vergleichbar mit einer Mailbox mit einer Größe von 0.

Der grundlegende Unterschied ist, dass der Sende-Task den Empfangs-Task direkt anspricht (adressiert über Handle). Ein Empfangs-Task kann jedoch nicht entscheiden von welchen Tasks er Daten bekommt. Mit Mailboxen kann jeder Task jede Mailbox verwenden. Jedoch kann ein Sende-Task nicht ermitteln wer die Daten erhält und der Empfangs-Task weiß nicht wer die Daten geschickt hat.

Ein weiterer Unterschied ist, dass es keinen Datenpuffer gibt. Bevor die Daten kopiert werden können, muss der Sende- und Empfangs-Task für die Übergabe bereit sein. Wenn zwei Tasks mittels Message-Passing Daten austauschen wollen, wird deshalb der erste Task blockiert, sobald dieser den Sende- oder Empfangsstatus erreicht, bis der zweite Task seinen entsprechenden Empfangs- oder Sendestatus erreicht hat. Die Datenübergabe wird dann abgeschlossen und beide Tasks können fortgesetzt werden.

Mailboxen stellen eine nicht so enge Verbindung zur Verfügung. Zum Beispiel kann ein Task sofort nach einer Operation weiter ausgeführt werden, auch wenn es keinen Task gibt der Daten empfangen kann. Message-Passing kann auch ausschließlich für die Synchronisation verwendet werden. In diesem Fall legt der Empfangs-Task eine Datenlänge von 0 fest und der Zeiger auf die Daten kann ein Wert von NULL haben, weil LasalOS eine Datenübertragung nicht ausführt.

Aufgrund der engen Kopplung zwischen Tasks und dem fehlenden Datenpuffer, wird dieser Mechanismus in der Regel nicht mit Interrupt-Handler verwendet. Dies ist jedoch möglich.

Beachten Sie, dass bei Mailboxen die verwendete Mailbox als Parameter übergeben werden muss. Beim Message-Passing legt der Sende-Task den Empfänger (Task-Handle) als Parameter fest. Da der Empfangs-Task nicht weiß, wer die Daten geschickt hat, legt er die Quelle nicht fest.

2.2.1.4 Semaphore Handhabung

Semaphoren sind beliebte Werkzeuge zur Synchronisierung von Tasks. Ein Semaphor kann als ein Ereigniszähler betrachtet werden, der niemals einen negativen Wert erhalten kann und von jedem Task mit den in diesem Abschnitt beschriebenen Funktionen verwendet werden kann. Ein Programm kann eine beliebige Anzahl von Semaphoren verwenden.

Die SIGNAL-Funktion speichert ein Event in einem Semaphor, WAIT ruft dann einen Event von dem Semaphor ab, wenn es zur Verfügung steht. Wenn kein Event zur Verfügung steht wird gewartet bis ein Event eingetreten ist. Es gibt fünf Arten von Semaphoren im LASAL OS: Zähler, Binär-, Event-, Ressourcen- und Mutex- Semaphoren. Der Semaphor -Typ wird mit der Erstellung des Semaphors definiert.

Zähler- Semaphoren können bis zu $2^{32}-1$ Events speichern und sind nützlich zur Synchronisation und entspricht der Dijkstra oder Ben-Ari Definitionen. Binär-Semaphoren können nicht als Zähler verwendet werden. Die SIGNAL-Funktion setzt immer ein binäres Semaphor auf 1, obwohl sie bereits den Wert von 1 erhalten hat. Die WAIT-Funktion setzt den Wert auf 0.

Event Semaphor sind ähnlich wie binäre Semaphoren mit der Ausnahme, dass die WAIT Funktion den Zählerwert nicht dekrementiert. Ein einziger Aufruf der SIGNAL-Funktion kann eine beliebige Anzahl von Tasks auf Ready setzen, die auf das Event- Semaphor warten. Um ein Event-Semaphor auf 0 zurückzusetzen, muss die RESETEVENT oder PULSE-Funktion aufgerufen werden. Diese Funktionen sind nur für Event- Semaphoren verfügbar. Die SIGNAL-Funktion setzt einen Event- Semaphor immer auf den Wert 1; die WAIT Funktion hat keinen Einfluss auf den Wert des Event-Semaphoren.

Ressourcen-Semaphoren sind für Ressourcenmanagement. Ein Ressourcen-Semaphor sorgt dafür, dass die Priorität eines Ressourcen-Semaphor belegenden Tasks mindestens die höchste Priorität aller anderen Tasks hat, welche die entsprechende Ressource verlangt. Diese Technik wird Prioritätsvererbung genannt. Ein Task, der eine Ressource mittels der Wait-Funktion beantragt, gibt seine Priorität an den Task der belegenden Ressource weiter. Dieser stellt sicher, dass ein Task mit hoher Priorität nie durch einen niedriger priorisierten Task unnötig blockiert ist.

Prioritätsvererbung kann man als Pseudo-Prioritäten für Ressourcen-Semaphoren ansehen. Zum Beispiel: Ein Task möchte mittels der Funktion WAIT ein Ressource-Semaphor belegen, wird aber blockiert, da die Ressource belegt ist. Der Task übergibt seine Priorität an den Ressource-Semaphor, der diese dann an den Ressource-belegenden Task übergibt. Eine Priorität kann nur vererbt werden, wenn die Priorität des Ressourcen-belegenden Tasks niedriger als die des wartenden Tasks ist. Die Priorität eines Tasks kann nur erhöht werden; diese kann niemals niedriger als die Basispriorität sein. Ressourcen Semaphoren ermöglichen auch die sichere Beendigung und

Suspendierung der Tasks. Ein Task kann nur sofort suspendiert oder beendet werden, wenn dieser keine Ressource-Semaphor besetzt. Ansonsten läuft er weiter, bis er alle Ressourcen freigegeben hat und später sich suspendiert oder beendet. Dieses Tool dient zur Vermeidung von Deadlocks. Bei Ressourcen Semaphoren gelten zwei wichtige Einschränkungen: Eine Ressource muss immer mittels der SIGNAL-Funktion durch den Task freigegeben werden, der diese durch die WAIT-Funktion erworben hat.

Ein Task kann eine beliebige Anzahl von Ressourcen gleichzeitig besetzen. Jedoch müssen die Ressourcen, in der genau umgekehrten Reihenfolge, wie sie angefordert wurden, wieder freigegeben werden.



```
OS_MT_Wait(R1);  
OS_MT_Wait(R2);  
...  
OS_MT_Signal(R2);  
OS_MT_Signal(R1);
```

Wenn die Reihenfolge der OS_MT_Signal Aufrufe ausgetauscht werden, wird ein Fehler im LasalOS auftreten.

Aufgrund der Einschränkungen sind Ressourcen-Semaphoren nicht für den Einsatz von Interrupt-Handler geeignet.

Mutex-Semaphoren sind eine Variation von Ressourcen-Semaphoren. Der einzige Unterschied besteht darin, dass ein Task eine Ressource, die es bereits besitzt, noch einmal anfordern kann.



```
OS_MT_Wait(R);  
OS_MT_Wait(R);  
...  
OS_MT_Signal(R);  
OS_MT_Signal(R);
```

Ist R ein Ressourcen-Semaphor, führt der zweite WAIT-Aufruf zu einem Fehler, da ein Task keine Ressource erwerben kann, die er bereits besitzt. Für ein Mutex-Semaphor ist dieses Konstrukt jedoch legal und die Ressource wird nicht freigegeben, bis die Zahl der SIGNAL-Funktionsaufrufe gleich wie denen der WAIT-Funktion ist. Der Nachteil von Mutex-Semaphoren im Vergleich mit Ressourcen-Semaphoren ist, dass ungleiche WAIT/SIGNAL Paare nicht erkannt werden. Wenn ein Aufruf an die SIGNAL-Funktionen fehlt, wird die Ressource nicht freigegeben und kein Fehler gemeldet, auch wenn der Task WAIT-/SIGNAL-Funktion an die gleiche Mutex weiter aufruft.

2.2.1.5 Mailbox Handhabung

Der vorherige Abschnitt beschrieb Semaphoren, die verwendet werden können um Tasks zu synchronisieren und stellt ein Mechanismus zur ordentlichen Inter-Task-Kommunikation mit globalen Daten zur Verfügung. Die Task-Kommunikation über globalen Daten ist jedoch schwer nachzuverfolgen und fehleranfällig, weil Tasks die erforderlichen Semaphor-Operationen vor dem Datenzugriff "vergessen" können. Darüber hinaus wird kein Protokoll zum kontrollierten Datenaustausch geführt. Mailboxen, die Datenpuffer sind, welche eine feste Anzahl von Nachrichten speichern können, werden zur Schließung dieser Lücke verwendet.

Im LasalOS können Nachrichten jede Größe haben und die Mailbox-Größe entsprechend konfiguriert werden. Tasks können Nachrichten in einer Mailbox speichern. Ist die Mailbox voll wird der Task blockiert, bis wieder Platz verfügbar ist. Tasks können auch Nachrichten von Mailboxen abrufen. Ist die Mailbox in diesem Fall leer, wird der Task ebenfalls blockiert. Die gleiche Mailbox kann verwendet werden, um Nachrichten von unterschiedlichen Tasks zu speichern bzw. abzurufen. Nachrichten können in einer Mailbox mittels [PUT\(\)](#), [PUTCOND\(\)](#) und [PUTTIMED\(\)](#) Funktionen zugefügt werden.

Nachrichten können auch am Anfang einer Nachrichtenkette mit den [PUTFRONT\(\)](#), [PUTFRONTCOND\(\)](#) und [PUTFRONTTIMED\(\)](#) Funktionen eingefügt werden. Sind Nachrichten gespeichert oder aus einer Mailbox durch den Aufruf der [PUT\(\)/GET\(\)](#) Funktionspaaren abgerufen worden, verhält sich die Mailbox als ein FIFO-Puffer (first in, first out). Dies bedeutet, dass Nachrichten aus einer Mailbox in der gleichen Reihenfolge wie sie gelagert waren abgerufen werden. Sind allerdings [PUTFRONT\(\)/GET\(\)](#) Funktionspaare aufgerufen, wird sich die Mailbox so verhalten, wie ein LIFO-Puffer (last in, first out). Für ein Höchstmaß an Flexibilität können die [PUT\(\)](#) und [PUTFRONT\(\)](#) Funktionen frei kombiniert werden (auch für dieselbe Mailbox).

2.2.2 Multitask-Klasse: Interface-Funktionen

2.2.2.1 Funktionen

Erweitert

GETLASTERROR	GETTIME		
------------------------------	-------------------------	--	--

Task-Funktionen

CREATETHREAD	CURRENTTASKHANDLE	DELAYUNTIL	GETMINSTACK
------------------------------	-----------------------------------	----------------------------	-----------------------------

GETTASKPRIORITY	GETTASKSTACK	GETTASKSTATE	RECEIVE
RECEIVECOND	RECEIVETIMED	RESUME	SEND
SENDCOND	SENDTIMED	SETPRIORITY	SUSPEND
TASKDELAY	TERMINATETASK		

Semaphoren

CREATESEMAPHORE	DELETESEMAPHORE	PULSE	RESETEVENT
RESOURCEOWNER	SEMAVALUE	SIGNAL	WAIT
WAITCOND	WAITTIMED		

Mailboxes

CLEARMAILBOX	CREATEMAILBOX	DELETEMAILBOX	GET
GETCOND	GETTIMED	MESSAGES	NEXTCOND
PUT	PUTCOND	PUTFRONT	PUTFRONTCOND
PUTFRONTTIMED	PUTTIMED		

Header Files

lsl_st_mt.h			
-------------	--	--	--

2.2.2.2 Erweiterte Funktionen

2.2.2.2.1 GETLASTERROR

Gibt letzten Fehler-Code zurück.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETLASTERROR
VAR_OUTPUT
  ret0:      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
keine		
Rückgabeparameter	Typ	Beschreibung

ret0	DINT	Letzter Fehler-Code
------	------	---------------------



Die OS-Interface-Funktionen geben entweder einem Fehler-Code (DINT-Wert) oder NIL-Wert zurück. GETLASTERROR() liefert den letzten aufgetretenen Fehler-Code in einer Funktion zurück.

Die folgenden Codes sind implementiert:

MERROR_NONE	Kein Fehler
MERROR_NOMEM	nicht genug Speicherplatz, um diese Funktion durchzuführen, z.B. interner alloc fehlgeschlagen.
MERROR_NOFCT	CreateTask braucht eine gültige Task-Funktion.
MERROR_FCTNOTINMEM	CreateTask braucht eine Task-Funktion platziert im LASAL Codespeicher.
MERROR_WRONGPRIOR	CreateTask unterstützt nur Prioritäten von 1 bis 14.
MERROR_STACK	CreateTask unterstützt Stackgrößen bis 0x4000.
MERROR_NAME	CreateTask, CreateMailbox oder CreateSemaphore braucht einen gültigen Namen
MERROR_NAMEUSED	CreateTask, CreateMailbox oder CreateSemaphore -Name werden schon verwendet.
MERROR_HANDLE	handle OS-List => nicht gültig
MERROR_NOTALLOWED	MT-Funktion wird in diesem TaskContext nicht erlaubt.
MERROR_MESSAGESIZE	mailbox braucht eine gültige Nachrichtengröße.
MERROR_DATA	Zeiger auf die Daten ist nicht gültig.

2.2.2.2.2 GETTIME

Diese Funktion kann verwendet werden, um die LASAL-OS Kernel-Uhr zu lesen.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETTIME
VAR_OUTPUT
    ret0:          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
-------------------	-----	--------------

keine		
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	Aktuelle Zeit der LasalOS internen Kernel-Uhr in Millisekunden

GETTIME liefert die Anzahl der Timer-Impulse seit dem Starten von LasalOS (wenn kein Überlauf aufgetreten ist).



2.2.2.3 Task-Funktionen

2.2.2.3.1 CREATETHREAD

Jeder Task in einem Programm kann andere Tasks/Threads mit dieser Funktion erstellen.

```
FUNCTION __CDECL VIRTUAL GLOBAL CREATETHREAD
VAR_INPUT
    taskfunction0      : pVoid;
    priority0         : UDINT;
    stackSize0        : UDINT;
    flags0            : UDINT;
    parameter0        : pVoid;
    name0             : ^char;
END_VAR
VAR_OUTPUT
    ret0              : MT_TASKHANDLE;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung						
taskfunction0	pVoid	Eine LASAL-Funktion mit __cdecl oder konventioneller LASAL CLASS Aufrufkonvention und einen einzigen pVoid Parameter (VAR_INPUT). es enthält den auszuführenden Code des neuen Task (Einsprungfunktion).						
priority0	UDINT	<p>Die Basispriorität des neuen Tasks als eine Ganzzahl zwischen MT_MIN_PRIORITY (1) und MT_MAX_PRIORITY (14).</p> <p>Die nachfolgende Tabelle gibt einen Überblick über die Standard-Tasks von LASAL OS</p> <table border="1"> <thead> <tr> <th>Thread (Aufgabenstellung)</th> <th>Aufgabenstellung</th> <th>die Priorität</th> </tr> </thead> <tbody> <tr> <td>Real-Time</td> <td>Ausführen des RtWork LASAL Tasks</td> <td>16</td> </tr> </tbody> </table>	Thread (Aufgabenstellung)	Aufgabenstellung	die Priorität	Real-Time	Ausführen des RtWork LASAL Tasks	16
Thread (Aufgabenstellung)	Aufgabenstellung	die Priorität						
Real-Time	Ausführen des RtWork LASAL Tasks	16						

		OS Kernel Task	Runtime-Überwachung, Ausführung Betriebssystem Funktionen	15
	Zyklisch	Ausführen des CyWork LASAL Tasks	14	
	Kommunikationsaufgaben	Online-Kommunikation	14	
	Background	Ausführung von Background LASAL Tasks Die Priorität des Background-Tasks hängt von der SET VISU LOW HIGH Einstellung ab (LOW: 10, HIGH: 14)	10/14	
stackSize0	UDINT	Größe des Stack, die für den neuen Task (in Bytes) angefordert wird; die Untergrenze liegt bei 512 Byte, die Obergrenze liegt bei 16 KB (16384 Bytes / 0x4000).		
flags0	UDINT	Kann verwendet werden, um Optionen für den neuen Task auszuwählen.		
parameter0	pVoid	Parameter, der an die Einsprungsfunktion des Tasks übergeben wird oder der this-Pointer (siehe flags0).		
name0	^char	Zeiger auf den Namen des Tasks; LasalOS fügt vor dem Namen USR_ ein.		
Rückgabeparameter	Typ	Beschreibung		
ret0	MT_TASKHANDLE	LASAL MT Task handle (MT_TASKHANDLE); Es handelt sich um eine Referenz auf den neu erstellten Task.		

Bei taskfunction0 (Einsprungsfunktion) kann zum Erstellen von Tasks/Threads mehrmals die gleiche Funktion verwendet werden. Diese Tasks werden dann den gleichen Code ausführen, jeder wird aber seine eigenen lokalen Daten haben. Tasks/Threads, die mit [CREATETHREAD\(\)](#) erstellt werden, werden immer nur einmal aufgerufen. Kommt die Ausführung an das Ende der Funktion, wird der Thread gelöscht (läuft aus). Um in dem Thread zu bleiben, muss also eine Schleife (versatile-Loop) implementiert werden, die dann immer wieder durchlaufen wird (Achtung: Abbruchbedingung formulieren – siehe [TERMINATETASK\(\)](#)).

Die Task-Priorität (priority0) gibt die Dringlichkeit an. Je höher diese ist, desto bevorzugter wird der Task ausgeführt. Wenn die Priorität des neuen Tasks höher ist als die des aktiven Tasks, wird der neue Task sofort aktiviert (wenn nicht MT_TASK_SUSPENDED als Flag gesetzt ist). LasalOS unterscheidet zwischen der Basispriorität und der Ausführungsriorität. Für das Scheduling (Taskwechsel), ist nur die Ausführungsriorität

interessant. Die Ausführungsriorität eines Tasks ergibt sich aus der Höhe seiner Basispriorität und die Prioritäten aller (Ressource- und Mutex-) Semaphoren, die durch den Task belegt sind. Die Priorität einer (Ressource- oder Mutex-) Semaphor ergibt sich aus der höchsten Priorität der Tasks, die auf diesen Semaphor warten. Diesen Vorgang nennt man Prioritätenvererbung (priority inheritance).

LasalOS benötigt mindestens 512 Byte Stack pro Thread. Ist der Parameter stackSize0 kleiner als 512, wird 512 (Byte) verwendet. Es ist zu beachten, dass Stack-Overflows zu den häufigsten und am schwersten feststellbaren Fehlern in Multitasking-Systemen gehören. Bei der Programmierung, sollten die [GETTASKSTACK\(\)](#) and [GETMINSTACK\(\)](#) Funktionen zum Analysieren der aktuellen Stack-Ausnutzung der Tasks umfassend verwendet werden. Bei der Entwicklungsphase sollte der Speicherplatz für die Stacks großzügig bemessen werden (8-16).

Die folgenden Werte sind derzeit für die Parameter semaphore s0 definiert:

MT_TASK_SUSPENDED	Ist dieses Flag angegeben, wird der Task im suspendierten Zustand erstellt. Bis RESUME() aufgerufen wird, wird es nicht gestartet.
MT_TASK_MATH_CONTEXT	Dieses Flag weist LASAL OS an, einen Gleitkommakontext für den neuen Task zu behalten. Dieses Flag kann nicht mit TF_NO_MATH_CONTEXT kombiniert werden.
MT_TASK_NO_MATH_CONTEXT	Dieses Flag weist LASAL OS an, einen Gleitkommakontext für den neuen Task nicht zu behalten. Der Flag kann nicht mit TF_MATH_CONTEXT kombiniert werden.
MT_TASK_SAVETHIS	Dieses Flag weist LASAL OS an, das ESI-Register mit dem Wert in parameter0 vor dem Ausführen des Tasks zu initialisieren. parameter0 muss den This-Pointer enthalten.

Wenn weder **MT_TASK_MATH_CONTEXT** noch **MT_TASK_NO_MATH_CONTEXT** angegeben sind, erstellt LasalOS standardmäßig einen Gleitkommakontext. Der Parameter, parameter0, wird an die Funktion des Tasks übergeben. LASAL OS interpretiert diesen Wert nicht, er wird einfach weitergegeben. Er kann verwendet werden, um beliebige Informationen an den neuen Task zu übergeben.

Der neue Task-Name (name0) ist zur einfachen Identifizierung und sollte nicht länger als 15 Zeichen sein (und darf nicht länger als 32 sein). LasalOS fügt, **USR_** am Anfang des Namen-Strings ein. Der Name darf nur einmal verwendet werden.

2.2.2.3.2 CURRENTTASKHANDLE

Gibt den Handle des aktuell ausgeführten Tasks zurück.

```
FUNCTION __cdecl virtual global currenttaskhandle
VAR_OUTPUT
    ret0      : MT_TASKHANDLE;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
keine			
Rückgabeparameter			Typ
ret0		MT_TASKHANDLE	Handle des aktuell laufenden Tasks oder NULL, wenn die Funktion in der LASAL TASK Ebene aufgerufen wird: RTWork, CyWork, usw. ..

2.2.2.3.3 DELAYUNTIL

Tasks, die zu einem bestimmten Zeitpunkt weiterlaufen sollen, können DELAYUNTIL() verwenden.

```
FUNCTION __CDECL VIRTUAL GLOBAL DELAYUNTIL
VAR_INPUT
    timeout0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
timeout0		UDINT	Zeit zum Weiterlaufen des Tasks in Millisekunden
Rückgabeparameter			Typ
ret0		DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code

DELAYUNTIL() kann verwendet werden, um zyklische Tasks umzusetzen, die in einem festen Zeitrahmen laufen.

2.2.2.3.4 GETMINSTACK

Liefert die Anzahl der am wenigsten verfügbaren Byte auf dem Stack eines Tasks seitdem dieser erstellt wurde.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETMINSTACK
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung

handle0	MT_TASKHANDLE	Handle des Tasks, dessen Stack abzufragen ist
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	Minimal verfügbarer Platz des Stacks oder -1, wenn ein Fehler aufgetreten ist

2.2.2.3.5 GETTASKPRIORITY

Liefert die aktuelle Ausführungsriorität eines Tasks zurück.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETTASKPRIORITY
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_TASKHANDLE	Handle des Tasks, dessen Priorität abzufragen ist
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	Zwischen MIN_PRIORITY und MAX_PRIORITY, wenn kein Fehler aufgetreten ist oder -1 bei einem Fehler

2.2.2.3.6 GETTASKSTACK

Gibt den verfügbaren Stack eines Tasks zurück.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETTASKSTACK
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_TASKHANDLE	Handle des Tasks, dessen Stack abzufragen ist
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	Freier Stack oder -1, wenn ein Fehler aufgetreten ist

Kann zur Abfrage des Stacks des aktuellen Tasks, **GETTASKSTACK([CURRENTTASKHANDLE\(\)](#))** verwendet werden. Es ist es möglich, dass LasalOS nicht in der Lage ist, die Stack-Grenzen zu ermitteln. In diesem Fall, wird der Wert **FFFFFFF** zurückgegeben. Der tatsächlich verfügbare Stack kann niemals diesen Wert erreichen.

2.2.2.3.7 GETTASKSTATE

Gibt den aktuellen Status eines Tasks zurück. Für den aktuellen Task kann das Task-Handle mit [CURRENTTASKHANDLE\(\)](#) geholt werden.

```
FUNCTION __cdecl virtual global GETTASKSTATE
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : LSL_MT_TASKSTATE;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_TASKHANDLE	Handle des Task, dessen Status zurückgegeben werden soll
Rückgabeparameter	Typ	Beschreibung
ret0	LSL_MT_TASKSTATE	Status vom Typ LSL_MT_TASKSTATE

Einer der folgenden Werte kann zurückgegeben werden:

MTASKSTATE_READY	Der Task ist bereit zum Ausführen.
MTASKSTATE_CURRENT	Der Task wird ausgeführt.
MTASKSTATE_SUSPENDED	Der Task wurde durch einen Aufruf von SUSPEND() suspendiert.
MTASKSTATE_DELAYING	Der Task wartet durch einen Aufruf von TASKDELAY() oder DELAYUNTIL() .
MTASKSTATE_BLOCKED_WAIT	Der Task wartet durch einen Aufruf von WAIT() auf einen Semaphor.
MTASKSTATE_TIMED_WAIT	Der Task wartet durch einen Aufruf von WAITTIMED() auf einen Semaphor.
MTASKSTATE_BLOCKED_PUT	Der Task wartet durch einen Aufruf von PUT() , bei einer vollen Mailbox.
MTASKSTATE_BLOCKED_GET	Der Task wartet durch einen Aufruf von GET() bei einer leeren Mailbox.

MTASKSTATE_TIMED_PUT	Der Task wartet durch einen Aufruf von PUTTIMED() bei einer vollen Mailbox.
MTASKSTATE_TIMED_GET	Der Task wartet durch einen Aufruf von GETTIMED() bei einer leeren Mailbox.
MTASKSTATE_TIMED_SEND	Der Task wartet durch einen Aufruf von SENDTIMED() auf den Empfänger-Task, dass dieser im Receive steht.
MTASKSTATE_BLOCKED_RECEIVE	Der Task wartet durch einen Aufruf von RECEIVE() um Daten von einem Sendetask zu erhalten (siehe SEND()).
MTASKSTATE_TIMED_SEND	Der Task wartet durch einen Aufruf von SENDTIMED() auf den Empfänger-Task, dass dieser im Receive steht.
MTASKSTATE_TIMED_RECEIVE	Der Task wartet durch einen Aufruf von RECEIVETIMED() um Daten von einem Sendetask zu erhalten (siehe SEND()).
MTASKSTATE_DEADLOCKED	Der Task ist durch eine Send-Operation blockiert (Message Passing), wobei der Empfänger-Task beendet wurde.
MTASKSTATE_ILLEGAL	Das übergebene Handle bezieht sich nicht auf einen bestehenden Task.
MTASKSTATE_TERMINATED	Der Task hat sich durch den Aufruf an TERMINATETASK() mit eigenem Handle beendet oder seine Task-Funktion abgeschlossen. Der Task kann nicht mehr laufen, aber besteht noch, weil sein Speicher noch nicht frei gegeben worden ist.

2.2.2.3.8 RECEIVE

Empfängt Daten von einem anderen Task.

```
FUNCTION __cdecl virtual global receive
VAR_INPUT
    data0          : PVOID;
    datalength0    : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
data0	pVoid	Zeiger auf die Variable, in der die empfangenen Daten gespeichert werden
datalength0	UDINT	Länge der zu erwartenden Daten
Rückgabeparameter	Typ	Beschreibung

ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist oder ein Fehler-Code
------	------	---------------------------------------------------------------------

Schickt ein Task Daten mit Hilfe von [SEND\(\)](#) oder [SENDTIMED\(\)](#) an einen in OS_MT_Receive() blockierten Empfangs-Task, so erfolgt die Datenübertragung sofort und der Task mit der höheren Priorität läuft weiter. Andernfalls wird der Empfangs-Task blockiert, bis ein anderer Task Daten sendet.

2.2.2.3.9 RECEIVECOND

Empfängt Daten von einem Task.

```
FUNCTION __CDECL VIRTUAL GLOBAL RECEIVECOND
VAR_INPUT
    data0          : PVOID;
    datalength0    : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
data0	pVoid	Zeiger auf die Variable, in der die empfangenen Daten gespeichert werden
datalength0	UDINT	Länge der zu erwartenden Daten
Rückgabeparameter	Typ	Beschreibung
ret0		TRUE, wenn die Datenübergabe erfolgreich abgeschlossen wurde, sonst ein Fehler-Code

Empfängt Daten von jedem Task unter der Bedingung, dass ein Task sofort zum Senden bereit ist. RECEIVECOND() führt nie zu einem Blockieren des Tasks.

2.2.2.3.10 RECEIVETIMED

Empfängt Daten von einem Task.

```
FUNCTION __CDECL VIRTUAL GLOBAL RECEIVETIMED
VAR_INPUT
    data0          : PVOID;
    datalength0    : UDINT;
    timeout0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
data0		pVoid	Zeiger auf die Variable, in der die empfangenen Daten gespeichert werden
datalength0		UDINT	Länge der zu erwartenden Daten
timeout0		UDINT	Wartezeit, bis der Empfangs-Task bereit ist um Daten zu akzeptieren (in Millisekunden)
Rückgabeparameter		Typ	Beschreibung
ret0		DINT	TRUE, wenn die Datenübergabe erfolgreich abgeschlossen wurde, sonst ein Fehler-Code

Empfängt Daten von jedem anderen Task, wenn ein Task bereit ist innerhalb der Wartezeit Daten zu senden.

2.2.2.3.11 RESUME

Reaktiviert einen suspendierten Task.

```
FUNCTION __CDECL VIRTUAL GLOBAL RESUME
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
handle0		MT_TASKHANDLE	Handle des zu deaktivierenden Tasks
Rückgabeparameter		Typ	Beschreibung
ret0		DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code

Ist der Task nicht suspendiert, hat RESUME() keine Wirkung.

2.2.2.3.12 SEND

Schickt Daten an einen anderen Task.

```
FUNCTION __CDECL VIRTUAL GLOBAL SEND
VAR_INPUT
    handle0      : MT_TASKHANDLE;
    data0        : pVoid;
END_VAR
VAR_OUTPUT
```

```
    ret0      : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
handle0	MT_TASKHANDLE	Handle des Empfangs-Tasks	
data0	pVoid	Zeigt auf die zu sendenden Daten	
Return parameters		Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code	

Wartet der Empfangs-Task in einem [RECEIVE\(\)](#) oder [RECEIVETIMED\(\)](#), erfolgt die Datenübertragung sofort und der Task mit der höheren Priorität läuft weiter. Andernfalls wird der Sende-Task blockiert, bis der Empfangs-Task bereit ist die Daten zu akzeptieren.

2.2.2.3.13 SENDCOND

Schickt Daten an einen anderen Task.

```
FUNCTION __CDECL VIRTUAL GLOBAL SENDCOND
VAR_INPUT
    handle0  : MT_TASKHANDLE;
    data0    : pVoid;
END_VAR
VAR_OUTPUT
    ret0    : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
handle0	MT_TASKHANDLE	Handle des Empfangs-Tasks	
data0	pVoid	Zeigt auf die zu sendenden Daten	
Rückgabeparameter		Typ	Beschreibung
ret0	DINT	TRUE, wenn die Datenübergabe erfolgreich abgeschlossen wurde, sonst ein Fehler-Code	

Daten werden an einen anderen Task geschickt, unter der Bedingung, dass der Empfangs-Task bereit ist Daten sofort zu akzeptieren. Ansonsten gibt der Rückgabewert einen Fehler aus und es werden keine Daten übertragen. SENDCOND() führt nie zu einem Blockieren des Tasks.

2.2.2.3.14 SENDTIMED

Schickt Daten an einen anderen Task.

```
FUNCTION __CDECL VIRTUAL GLOBAL SENDTIMED
VAR_INPUT
    handle0      : MT_TASKHANDLE;
    data0        : pVoid;
    timeout0     : UDINT;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_TASKHANDLE	Handle des Empfangs-Tasks
data0	pVoid	Zeigt auf die zu sendenden Daten
timeout0	UDINT	Wartezeit, bis der Empfangs-Task bereit ist um Daten zu akzeptieren (in Millisekunden)
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	TRUE, wenn die Datenübergabe erfolgreich abgeschlossen wurde, sonst ein Fehler-Code

Daten werden an einen anderen Task geschickt, wenn der Empfangs-Task innerhalb der Wartezeit bereit ist Daten zu akzeptieren.

2.2.2.3.15 SETPRIORITY

Ändern der Priorität eines Tasks. Die neue Priorität muss zwischen 1 und 14 liegen

```
FUNCTION __CDECL VIRTUAL GLOBAL SETPRIORITY
VAR_INPUT
    handle0      : MT_TASKHANDLE;
    priority0    : UDINT;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_TASKHANDLE	Handle des Task, dessen Priorität zu ändern ist
priority0	UDINT	Neue Priorität der Aufgabe
Rückgabeparameter	Typ	Beschreibung

ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code
------	------	----------------------------------------------------------------------

Priority0 muss im Bereich MIN_PRIO (1) bis MAX_PRIO (14) sein. Nach einem Aufruf auf OS_MT_SetPriority(), wird die Ausführungsriorität des Tasks neu bewertet und wenn nötig, führt der Scheduler einen Task-Wechsel aus.

2.2.2.3.16 SUSPEND

Kann verwendet werden, um einen Task zu deaktivieren.

```
FUNCTION __CDECL VIRTUAL GLOBAL SUSPEND
VAR_INPUT
    handle0 : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0 : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_TASKHANDLE	Handle des zu deaktivierenden Tasks
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code

Nach der Suspendierung des Tasks kann dieser nur durch den Aufruf der Resume-Funktion reaktiviert werden. Ist der Task bereits suspendiert hat die SUSPEND()-Funktion keine Wirkung. Vor der tatsächlichen Suspendierung des Tasks sorgt LasalOS dafür, dass der Task keine Ressource- oder Mutex-Semaphor belegt. Wenn ja, läuft der Task weiter, bis alle Ressourcen freigegeben wurden. Der Aufruf von SuspendTask wartet nicht bis die Suspendierung abgeschlossen ist, sondern läuft sofort weiter.

2.2.2.3.17 TASKDELAY

Blockiert den aufgerufenen Task für die angegebene Zeit und ermöglicht die Ausführung anderer Tasks.

```
FUNCTION __CDECL VIRTUAL GLOBAL TASKDELAY
VAR_INPUT
    timeout0 : UDINT;
END_VAR
VAR_OUTPUT
    ret0 : DINT;
END_VAR;
```

Übergabeparameter		
timeout0	UDINT	Bestimmt die Zeit in Millisekunden, die der Task blockiert werden muss
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code

If `TASKDELAY()` is called with `timeout0 = 0`, LasalOS Kernel checks whether other tasks with the same or a higher priority are ready. Wenn ein Task im State `MT_Ready` (bereit) mit höherer Priorität gefunden wird, wird dieser aktiviert. Sind Tasks mit der gleichen Priorität bereit, wird der Task mit der längsten Wartezeit aktiviert.

Somit kann `TASKDELAY(0)` verwendet werden, um Round-Robin-Scheduling (auch als kooperatives Time-Slicing bekannt) zu implementieren. Time-Slicing muss nicht zu diesem Zweck aktiviert werden.

2.2.2.3.18 TERMINATETASK

Beendet einen Task.

```
FUNCTION __cdecl virtual global TERMINATETASK
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter		
handle0	MT_TASKHANDLE	Handle des zu beendenen Tasks
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code

Die `OS_MT_TERMINATETASK` Funktion ist normalerweise nicht erforderlich. Ein Task wird beendet, wenn er das Ende seiner Einsprungsfunktion erreicht. Wird die Applikation beendet (Reset, Error, ...) werden auch alle User-Tasks beendet (Threads, die mit [CREATETHREAD\(\)](#) angelegt wurden).

Wenn ein ungültiger Wert an die Funktion übergeben wird, liefert das Programm einen Fehler. Bevor der Task beendet wird, muss LasalOS sichergehen, dass der Task kein anderen Ressource- oder Mutex-Semaphor besetzt. Ist dies der Fall, läuft der Task

weiter, bis alle Ressourcen freigegeben wurden. Die TERMINATETASK() Funktion wartet nicht bis der Task beendet ist; sondern gibt direkt zurück.

Die Ressourcen (Stack usw.) des zu beendeten Tasks werden frei gegeben, vorausgesetzt "taskhandle0" referenziert nicht den aktuellen Task. Der Speicher eines selbstbeendeten (nicht mit TERMINATETASK() beendet, sondern ausgelaufen) Tasks wird beim nächsten Aufruf von [CREATETHREAD\(\)](#) freigegeben. Bis dahin existiert der Task noch im terminierten Zustand, wird aber nicht mehr ausgeführt. Es ist zu vermeiden, dass ein Task zweimal beendet wird (selbstbeendet und durch einen anderen Task terminiert), da der Tasks beim zweiten Aufruf von TERMINATETASK() nicht mehr existiert.

2.2.3 Semaphor-Funktionen

2.2.3.1 CREATESEMAPHORE

Erstellt und initialisiert ein Semaphor.

```
FUNCTION __CDECL VIRTUAL GLOBAL CREATESEMAPHORE
VAR_INPUT
    type0      : LSL_MT_SEMATYPE;
    init0     : UDINT;
    flags0    : UDINT;
    name0     : ^char;
END_VAR
VAR_OUTPUT
    ret0      : MT_SEMAHANDLE;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
type0	LSL_MT_SEMATYPE	Gewünschter Semaphor-Typ
init0	UDINT	Muss für den gewählten Semaphor-Typ gültig sein
flags0	UDINT	Gibt an, ob und wie die Funktion nach einem bestehenden Semaphor suchen sollte
name0	^char	Zeiger auf den Namen des Semaphor
Rückgabeparameter	Typ	Beschreibung
ret0	MT_SEMAHANDLE	Gültiger Handle oder NIL, wenn die Funktion fehlerhaft ist



Die Werte für den Parameter type0:

- MTSEMATYPE_COUNTING
- MTSEMATYPE_BINARY
- MTSEMATYPE_EVENT

-
- MTSEMATYPE_RESOURCE
 - MTSEMATYPE_MUTEX

Für Counting-Semaphoren muss der Parameter init0 0 bis $2^{32}-1$ sein, für Binary- und Event-Semaphoren 0 oder 1 und für Ressource/Mutex Semaphoren muss dieser Parameter 1 sein.

Die Werte für den Parameter flags0:

Der Wert 0: Ein neuer Semaphor wird ohne eine Bedingung erstellt.

MTSEMACREATE_SEARCH CREATESEMAPHORE versucht ein Semaphor von dem gleichen type0 und name0 zu finden, der auch mit MTSEMACREATE_SEARCH erstellt wurde. Unbenannte Semaphoren werden nicht geprüft und bei dem Vergleich der Namen wird zwischen Groß- und Kleinschreibung unterschieden. Wenn ein Semaphor gefunden wurde, wird ein Handle auf diesem bestehenden Semaphor zurückgegeben und kein neues erstellt. In diesem Fall wird Parameter init0 ignoriert.

MTSEMACREATE_FAIL_NOT_FOUND

Dieses Flag kann zusätzlich zu MTSEMACREATE_SEARCH angegeben werden. Es wird, wenn kein Semaphor gefunden wurde, kein neues angelegt. In diesem Fall wird der Wert NIL in ret0 zurückgeliefert.

Der Name des neuen Semaphors ist nur zur einfachen Identifizierung des Semaphors und sollte nicht länger als 15 Zeichen sein. LASAL OS fügt vor dem Namen USR_ ein.

2.2.3.2 DELETESEMAPHORE

Ein bestehendes Semaphor wird gelöscht und ungültig.

```
FUNCTION __cdecl virtual global deletesemaphore
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_SEMAHANDLE	Handle auf das Semaphor
Rückgabeparameter	Typ	Beschreibung

ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code
------	------	----------------------------------------------------------------------



Kein Task darf auf das Semaphor warten. Der Versuch auf ein Semaphor zuzugreifen, nachdem es gelöscht wurde, führt zu einem Fehler und das Verhalten ist undefiniert.

2.2.3.3 PULSE

Stellt alle Tasks auf den Zustand Bereit (MT_Ready), die auf den Event-Semaphor warten, und setzt den Semaphor sofort zurück.

```
FUNCTION __CDECL VIRTUAL GLOBAL PULSE
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_SEMAHANDLE	Handle auf das Semaphor
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist oder ein Fehler-Code



Sollten ein oder mehrere dieser Tasks eine höhere Priorität als der aufrufende Task haben, wird ein Task-Wechsel gemacht. Bezieht sich handle0 nicht auf ein Event- Semaphor wird ein Fehler ausgelöst.

2.2.3.4 RESETEVENT

Setzt den Wert eines Event-Semaphor auf 0.

```
FUNCTION __CDECL VIRTUAL GLOBAL RESETEVENT
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
handle0		MT_SEMAHANDLE	Handle auf den Semaphor
Rückgabeparameter		Typ	Beschreibung
ret0		DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code



Nachfolgende Aufrufe von [WAIT\(\)](#) oder [WAITTIMED\(\)](#) sind blockierend. Um ein Event-Semaphor auf 1 zu setzen, verwenden Sie [SIGNAL\(\)](#). Um alle Tasks freizugeben die auf den Event-Semaphor warten, ohne den Semaphor zu setzen, verwenden Sie [PULSE\(\)](#).

2.2.3.5 RESOURCEOWNER

Kann verwendet werden, um festzustellen, welcher Task derzeit eine Ressourcen- oder Mutex-Semaphor belegt.

```
FUNCTION __CDECL VIRTUAL GLOBAL RESOURCEOWNER
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : MT_TASKHANDLE;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
handle0		MT_SEMAHANDLE	Handle des abzufragende Ressourcen- oder Mutex-Semaphor
Rückgabeparameter		Typ	Beschreibung
ret0		MT_SEMAHANDLE	Wurde das Semaphor freigegeben, ist der Rückgabewert NIL, ansonsten wird der Task-Handle zurückgegeben

2.2.3.6 SEMAVALUE

Die Anzahl der Events, die in einem Semaphor gespeichert sind, kann hiermit abgefragt werden.

```
FUNCTION __CDECL VIRTUAL GLOBAL SEMAVALUE
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
handle0		MT_SEMAHANDLE	Handle auf das Semaphor
Rückgabeparameter		Typ	Beschreibung
ret0		DINT	Zahl der Events oder ein Fehler-Code

Die Funktion führt nicht zu einem Task-Wechsel.



2.2.3.7 SIGNAL

Speichert ein Event in einem Semaphor.

```
FUNCTION __CDECL VIRTUAL GLOBAL SIGNAL
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
handle0		MT_SEMAHANDLE;	Handle des Semaphors, um das Event zu speichern
Rückgabeparameter		Typ	Beschreibung
ret0		DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code

Wenn mehrere Tasks auf das Semaphor warten, wird der Task mit der höchsten Priorität auf bereit gesetzt (MT_Ready). Ist seine Priorität höher als die des anrufenden Tasks, wird er sofort aktiviert (MT_Current). Ist der Task von Type MTSEMATYPE_EVENT, werden alle wartenden Tasks auf bereit gesetzt (MT_Ready). Wenn kein Task auf das Semaphor wartet, wird das Ereignis gespeichert. Eine Zähler-Semaphor kann bis zu $2^{32} - 1$ Ereignisse speichern. Die Applikation sollte dafür sorgen, dass diese Grenze nicht überschritten wird. Binär- und Event- Semaphoren ignorieren zusätzliche Ereignisse, wenn sie bereits eins erhalten haben. Der Versuch ein Ressource- oder Mutex-Semaphor auf einen Wert > 1 mit der Funktion SIGNAL zu setzen (mehrmals Signal aufrufen), ist ein Fehler. In diesem

Fall werden die Ergebnisse unvorhersehbar. Wenn semahandle0 eine Ressource oder Mutex Semaphor ist, wird die (Ausführungs-)Priorität des aktiven Tasks nach den Regeln der Prioritätsvererbung neu bewertet.

2.2.3.8 WAIT

Ruft einen Event aus einem Semaphor ab.

```
FUNCTION __CDECL VIRTUAL GLOBAL WAIT
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_SEMAHANDLE	Handle des Semaphor, das den Event erhält
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code



Wenn kein Ereignis verfügbar ist, wird der aufrufende Task blockiert (MT_BLOCKED_WAIT). Er kann nur reaktiviert werden, indem ein anderer Task die Funktion SIGNAL für das jeweilige Semaphor aufruft (MT_READY). Eine beliebige Anzahl von Tasks kann bei einem Semaphor auf Ereignisse warten. Die Tasks werden in der Reihenfolge ihrer Prioritäten abgearbeitet (MT_CURRENT).

Wenn semahandle0 eine besetzte Ressource- oder Mutex-Semaphor ist, wird die (Ausführungs-)Priorität des Tasks, der das Semaphor besetzt, auf die des blockierenden angehoben, wenn diese höher ist (Prioritätenvererbung).

Nach dem Aufruf von WAIT besitzt oder besetzt der aktive Task der Ressource- oder Mutex-Semaphor. Er kann nicht ausgesetzt oder beendet werden, bis er alle seine Ressourcen freigegeben hat.

Eine Ausnahme bildet Event-Semaphoren. Sie dekrementieren mit WAIT den Wert des Semaphors nach Abschluss der Operation.

2.2.3.9 WAITCOND

Ruft ein Ereignis aus einem Semaphor ab, wenn eins vorhanden ist.

```
FUNCTION __CDECL VIRTUAL GLOBAL WAITCOND
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_SEMAHANDLE	Handle des Semaphor, das den Event erhält
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	TRUE, wenn ein Signal abgerufen wurde, ansonsten wird ein Fehler-Code zurückgeliefert

Ruft ein Event von einem Semaphor ab, wenn eines sofort verfügbar ist. WAITCOND() führt nie zu einem blockierenden Task-Wechsel.



2.2.3.10 WAITTIMED

Versucht ein Ereignis aus einem Semaphor abzurufen, solange kein Timeout auftritt.

```
FUNCTION __CDECL VIRTUAL GLOBAL WAITTIMED
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
    timeout0    : UDINT;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
handle0	MT_SEMAHANDLE	Handle des Semaphor, der den Event erhält
timeout0	UDINT	Maximale Wartezeit (in Millisekunden), bis ein Event auftritt
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	TRUE, wenn ein Event abgerufen wurde; andernfalls war kein Event verfügbar und Timeout ist abgelaufen



Wenn semahandle0 eine besetzte Ressource- oder Mutex-Semaphor ist, wird die (Ausführungs-)Priorität des Tasks, der die Semaphor besetzt, auf die des blockierenden angehoben, wenn diese höher ist (Prioritätenvererbung).

Nach erfolgreichem Abschluss der WAITTIMED() Funktion belegt/besitzt der aktive Task das Semaphor. Er kann nicht ausgesetzt oder beendet werden, bis er alle seine Ressourcen freigegeben hat.

Eine Ausnahme bildet Event-Semaphoren. Sie dekrementieren mit [WAIT\(\)](#) den Wert des Semaphors nach erfolgreichem Abschluss (ret0 = TRUE) der Operation.

2.2.4 Mailbox-Funktionen

2.2.4.1 CLEARMAILBOX

Löscht den Inhalt einer Mailbox.

```
FUNCTION __cdecl virtual global clearmailbox
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
mbhandle0	MT_MAILBOX	Handle der zu leerenden Mailbox
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code



Alle gespeicherten Daten in der Mailbox werden verworfen. Ist die Mailbox voll und ein Task wartet, um in die Mailbox zu schreiben, wird der Task von CLEARMAILBOX() wieder auf Bereit gestellt (MT_Ready).

2.2.4.2 CREATEMAILBOX

Erstellt und initialisiert eine Mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL CREATEMAILBOX
VAR_INPUT
    messagelen0      : UDINT;
    messageslots0    : UDINT;
    name0            : ^char;
END_VAR
VAR_OUTPUT
    ret0             : MT_MAILBOX;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
messagelen0		UDINT	Länge einer Mailbox-Nachricht in Byte
messageslots0		UDINT	Maximale Anzahl von Nachrichten, die die Mailbox speichern kann
name0		^char	Zeigt auf den Namen der Mailbox
Rückgabeparameter		Typ	Beschreibung
ret0		MT_MAILBOX	Referenz auf die neue Mailbox oder NIL, wenn ein Fehler aufgetreten ist



LasalOS fügt USR_ am Anfang des String-Namens ein. CREATEMAILBOX() reserviert und initialisiert die Mailbox.

2.2.4.3 DELETEMAILBOX

Gibt den Speicher einer Mailbox frei.

```
FUNCTION __CDECL VIRTUAL GLOBAL DELETEMAILBOX
VAR_INPUT
    mbhandle0       : MT_MAILBOX;
END_VAR
VAR_OUTPUT
    ret0            : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
mbhandle0		MT_MAILBOX	Handle der zu löschenende Mailbox
Rückgabeparameter		Typ	Beschreibung

ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code
------	------	----------------------------------------------------------------------



LasalOS prüft, ob ein Task auf eine Mailbox wartet. In diesem Fall wird das Programm abgebrochen. Die Applikation muss dafür sorgen, dass gelöschte Mailboxen nicht mehr verwendet werden.

2.2.4.4 GET

Ruft eine Nachricht aus einer Mailbox ab.

```
FUNCTION __cdecl virtual global GET
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0         : pVoid;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
mbhandle0	MT_MAILBOX	Handle der Mailbox, die die abzurufenden Nachrichten enthält
data0	pVoid	Zeigt auf die Variable, die die Nachricht speichert
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code



Ist die Mailbox leer, wird der aufrufende Task blockiert, bis ein anderer Task (oder Interrupt-Handler) eine Nachricht speichert. Ist die Mailbox voll und ein anderer Task wartet, um eine Nachricht in die Mailbox zu speichern, wird der wartende Task auf Bereit gestellt. Hat er eine höhere Priorität, wird er sofort aktiviert.

2.2.4.5 GETCOND

Ruft eine Nachricht aus einer Mailbox ab.

```
FUNCTION __cdecl virtual global PUTFRONTCOND
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0         : pVoid;
END_VAR
VAR_OUTPUT
```

```
    ret0      : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
mbhandle0		MT_MAILBOX	Handle der Mailbox, die die abzurufenden Nachrichten enthält
data0		pVoid	Zeigt auf die Variable, die die Nachricht speichert
Rückgabeparameter		Typ	Beschreibung
ret0		DINT	TRUE, wenn die Nachricht erfolgreich zurückgeholt wurde, sonst wird ein Fehler-Code zurückgeliefert



Ruft eine Nachricht von einer Mailbox ab, wenn eine sofort verfügbar ist. GETCOND() führt nie zu einem blockierenden Task-Wechsel. Ist die Mailbox leer, wird dies durch den Rückgabewert angegeben und es werden keine Daten übertragen.

2.2.4.6 GETTIMED

Ruft eine Nachricht aus einer Mailbox ab. Ist die Mailbox leer, wird eine bestimmte Zeitspanne abgewartet, bis eine Nachricht verfügbar wird.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETTIMED
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
    timeout0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
mbhandle0		MT_MAILBOX	Handle der Mailbox, die die abzurufenden Nachrichten enthält
data0		pVoid	Zeigt auf die Variable, die die Nachricht speichert
timeout0		UDINT	Wartezeit (in Millisekunden), bis eine Nachricht in der Mailbox verfügbar wird
Rückgabeparameter		Typ	Beschreibung
ret0		DINT	TRUE, wenn die Nachricht erfolgreich zurückgeholt wurde, sonst wird ein Fehler-Code zurückgeliefert

2.2.4.7 MESSAGES

Gibt die Anzahl der derzeit gespeicherten Nachrichten in einer Mailbox zurück.

```
FUNCTION __CDECL VIRTUAL GLOBAL MESSAGES
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
END_VAR
VAR_OUTPUT
    ret0          : UDINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
mbhandle0	MT_MAILBOX	Handle der abzufragenden Mailbox
Rückgabeparameter	Typ	Beschreibung
ret0	UDINT	Rückgabewert wird immer zwischen 0 und dem Wert des Parameters messageslots0 von CREATEMAILBOX() liegen, als die Mailbox erstellt wurde

2.2.4.8 NEXTCOND

Die nächste Nachricht wird abgefragt.

```
FUNCTION __CDECL VIRTUAL GLOBAL NEXTCOND
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
mbhandle0	MT_MAILBOX	Handle der Mailbox, die die abzufragende Nachricht enthält
data0	pVoid	Zeigt auf die Variable, die die Nachricht speichert
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	TRUE, wenn mindestens eine Nachricht in der Mailbox verfügbar ist und data0 enthält eine Kopie der Nachricht, sonst wird ein Fehler-Code zurückgeliefert



Fragt die nächste Nachricht von einer Mailbox ab, wenn mindestens eine Nachricht verfügbar ist. Aber im Gegensatz zu [GETCOND\(\)](#), wird die Nachricht nicht abgerufen. NEXTCOND() führt nie zu einem blockierenden Task-Wechsel.

2.2.4.9 PUT

Speichert eine Nachricht in eine Mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUT
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
mbhandle0	MT_MAILBOX;	Handle der Mailbox, in der die Nachricht gespeichert wird
data0	pVoid	Wird in der Mailbox gespeichert
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code



Ist die Mailbox voll, wird der aufrufende Task blockiert, bis ein anderer Task (oder Interrupt-Handler) eine Nachricht abruft. Ist die Mailbox leer und ein anderer Task wartet bei der Mailbox auf eine Nachricht, wird die wartende Task auf Bereitgestellt. Hat er eine höhere Priorität, wird er sofort aktiviert.

2.2.4.10 PUTCOND

Speichert eine Nachricht in eine Mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTCOND
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
mbhandle0	MT_MAILBOX	Handle der Mailbox, in der die Nachricht gespeichert wird
data0	pVoid	Wird in der Mailbox gespeichert
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	TRUE, wenn die Nachricht erfolgreich gespeichert wurde, sonst wird ein Fehler-Code zurückgegeben.



Speichert eine Nachricht in eine Mailbox, wenn genügend Platz verfügbar ist. PUTCOND() führt nie zu einem blockierenden Task-Wechsel. Ist die Mailbox voll, wird dies durch den Rückgabewert angegeben und es werden keine Daten übertragen.

2.2.4.11 PUTFRONT

PUTFRONT() entspricht [PUT\(\)](#), fügt aber die Nachricht am Beginn der Mailbox-Warteschlange ein, anstatt am Ende.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTFRONT
VAR_INPUT
    mbhandle0    : MT_MAILBOX;
    data0        : pVoid;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
mbhandle0	MT_MAILBOX	Handle der Mailbox, in der die Nachricht gespeichert wird
data0	pVoid	Wird in der Mailbox gespeichert
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	MTERROR_NONE, wenn kein Fehler aufgetreten ist, oder ein Fehler-Code

2.2.4.12 PUTFRONTCOND

PUTFRONTCOND() entspricht [PUTCOND\(\)](#), fügt jedoch die Nachricht am Beginn der Mailbox-Warteschlange ein, anstatt am Ende.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTFRONTCOND
VAR_INPUT
```

```

mbhandle0      : MT_MAILBOX;
data0          : pVoid;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;

```

Übergabeparameter		Typ	Beschreibung
mbhandle0		MT_MAILBOX	Handle der Mailbox, in der die Nachricht gespeichert wird
data0		pVoid	Wird in der Mailbox gespeichert
Rückgabeparameter		Typ	Beschreibung
ret0		DINT	TRUE, wenn die Nachricht erfolgreich gespeichert wurde, sonst wird ein Fehler-Code zurückgegeben

2.2.4.13 PUTFRONTTIMED

PUTFRONTTIMED() entspricht [PUTTIMED\(\)](#), fügt jedoch die Nachricht am Beginn der Mailbox-Warteschlange ein, anstatt am Ende.

```

FUNCTION __CDECL VIRTUAL GLOBAL PUTFRONTTIMED
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
    timeout0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;

```

Übergabeparameter		Typ	Beschreibung
mbhandle0		MT_MAILBOX	Handle der Mailbox, in der die Nachricht gespeichert wird
data0		pVoid	Wird in der Mailbox gespeichert
timeout0		UDINT	Wartezeit (in Millisekunden), bis Platz in der Mailbox zur Verfügung steht
Rückgabeparameter		Typ	Beschreibung
ret0		DINT	TRUE, wenn die Nachricht erfolgreich gespeichert wurde, sonst wird ein Fehler-Code zurückgegeben

2.2.4.14 PUTTIMED

Speichert eine Nachricht in einer Mailbox. Ist die Mailbox voll, wird eine bestimmte Zeitspanne gewartet, bis Platz zur Verfügung steht.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTTIMED
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
    timeout0       : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
mbhandle0	MT_MAILBOX	Handle der Mailbox, in der die Nachricht gespeichert wird
data0	pVoid	Wird in der Mailbox gespeichert
timeout0	UDINT	Wartezeit (in Millisekunden), bis genügend Platz in der Mailbox zur Verfügung steht
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	TRUE, wenn die Nachricht erfolgreich gespeichert wurde, sonst wird ein Fehler-Code zurückgegeben

2.2.5 Schnittstellen-Funktionen

Wird die OS-Interface-Klasse `_MultiTask` nicht verwendet, muss die Header-Datei `lsl_st_mt.h` eingebunden werden, um die folgenden Funktionen verwenden zu können.

TASK-Funktionen

OS_MT_CreateThread	OS_MT_CurrentTaskHandle	OS_MT_Delay	OS_MT_DelayUntil
OS_MT_GetMinStack	OS_MT_GetTaskPrio	OS_MT_GetTaskStack	OS_MT_GetTaskState
OS_MT_Receive	OS_MT_ReceiveCond	OS_MT_ReceiveTimed	OS_MT_Resume
OS_MT_Send	OS_MT_SendCond	OS_MT_SendTimed	OS_MT_SetPriority
OS_MT_Suspend	OS_MT_TerminateTask		

Semaphor-Funktionen

OS_MT_CreateSemaphore	OS_MT_DeleteSemaphore	OS_MT_Pulse	OS_MT_ResetEvent
OS_MT_ResourceOwner	OS_MT_SemaValue	OS_MT_Signal	OS_MT_Wait
OS_MT_WaitCond	OS_MT_WaitTimed		

Mailbox-Funktionen

OS_MT_ClearMailbox	OS_MT_CreateMailbox	OS_MT_DeleteMailbox	OS_MT_Get
OS_MT_GetCond	OS_MT_GetTimed	OS_MT_Messages	OS_MT_NextCond
OS_MT_Put	OS_MT_PutCond	OS_MT_PutFront	OS_MT_PutFrontCond
OS_MT_PutFrontTimed	OS_MT_PutTimed		

Erweiterte Funktionen

OS_MT_GetLastError	OS_MT_GetTime		
------------------------------------	-------------------------------	--	--

Alle Funktionen haben die gleichen Parameter und Rückgabewerte wie die oben beschriebenen Funktionen der Klasse `_MultiTask`.

2.2.5.1 Beispiele

Einen Task erstellen

Einen Task erstellen aus der Cyclic work.

```
FUNCTION VIRTUAL GLOBAL MyClass::CyWork
VAR_INPUT
    EAX      : UDINT;
END_VAR
VAR_OUTPUT
    state    : UDINT;
END_VAR
VAR
    handle   : MT_TASKHANDLE;
END_VAR

// create the task
handle := OS_MT_CreateThread(#MyTask(), 14, 0x2000, 0, NIL, "MyTask");

// this would terminate the task, but task terminates itself
// OS_MT_TerminateTask(handle);

state:= READY;
END_FUNCTION //VIRTUAL GLOBAL MyClass::CyWork

FUNCTION __CDECL MyClass::MyTask
VAR_INPUT
    param0  : pVoid;
END_VAR
VAR
    count    : UDINT;
END_VAR

count := 0;

WHILE count < 100 DO
    OS_MT_Delay(10);    // let tasks with lower priority run
END_WHILE;

// task terminates after while loop

END_FUNCTION //__CDECL MyClass::MyTask
```

Eine Nachricht senden

Ein Task schickt die Daten direkt auf einen zweiten Task.

Task 1

...

```
Task 1 do some work with data1
...
// function blocks until task 2 is ready to receive data
// function does not block if task 2 is immediately ready to receive
// or if an error occurred
returnvalue := OS_MT_Send(taskhandle2, #data1);
if returnvalue <> MTERROR_NONE then
    ...
end_if;

// function never blocks
returnvalue := OS_MT_SendCond(taskhandle2, #data1);

// functions blocks until task2 is ready to receive or timeout occurred
// does not block if task 2 is immediately ready
returnvalue := OS_MT_SendTimed(taskhandle2, #data1, 500);
...

```

Task 2

```
...
// waiting for data from task 1
// function blocks until task 1 is ready to send
// does not block if task 1 is immediately ready to send or an error
// occurred
OS_MT_Receive(#data, sizeof(data));

returnvalue := OS_MT_ReceiveCond(#data, sizeof(data));

returnvalue := OS_MT_ReceiveTimed(#data, sizeof(data), 1000);
...

```

Semaphor Handhabung

Task 1 soll aktiviert werden, wenn Task 2 einen bestimmten Punkt erreicht hat.

Task 2

```
MySema : MT_SEMAHANDLE;

...
// create a semaphore and initialize with value 0
MySema := OS_MT_CreateSemaphore(MTSEMATYPE_BINARY, 0, 0, "MySema");
...
// stores an event (binary can only store 1 event) - sets value to 1
OS_MT_Signal(MySema);
```

Task 1

```
...  
// retrieves an event - sets value to 0  
OS_MT_Wait(MySema);  
// function will block forever if no task stores an event  
...
```

Mailbox Handhabung

Task 1 holt Daten von Task 2.

Task 1

```
data1    : DINT;  
...  
// retrieves the message from task 2  
// function blocks until task 2 stores a message  
OS_MT_Get(MyMailbox, #data1$^DINT);    // parameter is pvoid  
...
```

Task 2

```
data2    : DINT;  
MyMailbox      : MT_MAILBOX;  
...  
// create a mailbox with message size 0 and 1 slot  
MyMailbox := OS_MT_CreateMailbox(sizeof(DINT), 1, "MyMailbox");  
...  
data2 := ...;  
// store a message  
// if no slot is free, function blocks until another task retrieves a  
// message  
OS_MT_Put(MyMailbox, #data2$^DINT);    // param is pvoid  
...
```

2.3 API Application Heap

2.3.1 Übersicht

Der Application-Heap ist für die Speicheranforderungen der Applikation reserviert. Die Applikation verwendet die OS_SSR_Alloc, OS_SSR_ReAlloc und OS_SSR_Free Funktionen um den Heapspeicher zuzuteilen, neu zuzuteilen und freizugeben.

Die Größe des Heap hängt von der Plattform ab. Die Variablen _UserHeapTotalSize (AT% * M * 0024), _UserHeapUsedMem (AT% * M * 0028) und UserHeapFreeMem (* AT% M 002C *), die in der Rtos_variable.h Header-Datei definiert sind, können verwendet werden um die Gesamtgröße sowie die Menge des zugewiesenen und freien Speicherplatzes zu überwachen.

2.3.2 Debug-Heap verwenden

Einige Probleme, die auf den Programmierer zukommen können, sind das Überschreiben eines am Ende zugewiesenen Puffers, das Schreiben vor dem zugewiesenen Puffer und auslaufenden Speicher, die von fehlerhaften freien Zuweisungen verursacht wurden, nachdem sie nicht mehr benötigt werden. Der Debug-Heap stellt Tools zur Lösung dieser Art von Speicherzuweisungsproblemen zur Verfügung.

Wenn der Debug-Heap aktiviert ist, reserviert der Heap-Manager einen etwas größeren Block vom Speicher als angefordert wurde und liefert einen Pointer auf den Teil dieses Blocks. Der von den Debug-Heap Routinen zugeordnete zusätzliche Speicher wird für Buchhaltungsinformationen, für Pointer, die die Debug-Speicherblöcke verbinden und für kleine Puffer an beiden Seiten der Daten, um das Überschreiben der zugeteilten Region zu verhindern, verwendet.

Die Debug-Heap Funktionen können mit den folgenden CLI-Befehlen aktiviert werden:

```
SET DBGHEAP
```

Anzeige der aktuellen DBGHEAP-Einstellungen.

```
SET DBGHEAP CHECK_ALWAYS ON|OFF
```

ON: Der gesamte Applikations-Heap wird auf Integrität bei jedem Aufruf einer Heap-Funktion geprüft.

OFF: Nur der in OS_SSR_ReAlloc oder OS_SSR_Free angegebene Block wird auf eine andere DBGHEAP-Einstellung geprüft.

```
SET DBGHEAP ERR_OUTOFCMEM ON|OFF
```

ON: Das Betriebssystem löst eine AppMem-Fehlermeldung aus, wenn OS_SSR_Alloc oder OS_SSR_ReAlloc wegen eines Speichermaßnahmen fehlerhaft ist.

```
SET DBGHEAP REALLOC_NEW_PTR ON|OFF
```

ON: Die Funktion OS_SSR_ReAlloc reserviert immer einen neuen Block an einem anderen Speicherort als der ursprüngliche Block.

SET DBGHEAP CHECK_LIMIT_BEGIN ON|OFF

ON: Der Heap-Manager teilt einem größeren Block am Anfang ihre Daten zu, als für einen kleinen Puffer angefordert wurde, um das Überschreiben der zugeteilten Region vor Beginn der Region zu verhindern. Der Heap-Manager füllt diesen Puffer mit 0xFD.

SET DBGHEAP CHECK_LIMIT_END ON|OFF

ON: Der Heap-Manager teilt einem größeren Block am Ende ihre Daten zu, als für einen kleinen Puffer angefordert wurde, um das Überschreiben der zugeteilten Region nach dem Ende der Region zu verhindern. Der Heap-Manager füllt diesen Puffer mit 0xFD.

SET DBGHEAP FILL_FREE_BLOCK ON|OFF

ON: Der Heap-Manager füllt die freigegebenen Blöcke mit einem bekannten Wert (0xDD), um zu überprüfen, ob der freigegebene Speicher nicht weiterhin beschrieben wird.

SET DBGHEAP FILL_NEW_BLOCK ON|OFF

ON: Neue Blöcke sind mit 0xCD gefüllt, wenn sie zugeteilt werden.

SET DBGHEAP DBGHEAP STORE_IP ON|OFF

ON: Der Heap-Manager teilt einen größeren Bereich zu als angefordert, welcher die IP (Instruction pointer) Adresse vom Heap-Aufruf enthält. Im Falle eines AppMem-Fehlers, wird die IP-Adresse des fehlerhaften Blockes im System log eingetragen.

SET DBGHEAP CHECK_FREE_PTR ON|OFF

ON: Das Betriebssystem löst eine AppMem-Fehlermeldung aus, wenn OS_SSR_Free auf einen Speicherbereich zeigt, der bereits freigegeben wurde.

SET DBGHEAP ALL ON|OFF

ON: Aktiviert alle der oben genannten Heap-Debug Optionen.

OFF: Deaktiviert alle der oben genannten Debug-Heap Optionen

Zu beachten ist, dass die Aktivierung einer Debug-Heap Funktion die Leistung ihrer Anwendung vermindern könnte. Die Debug-Heap Funktionen stehen seit LasalOS 5.52 zur Verfügung.

2.3.3 SSR-Interface Funktionen für den Heap

Header Dateien: Isl_st_ifssr.h

2.3.3.1 OS_SSR_Malloc

Die OS_SSR_Malloc Funktion reserviert einen Speicherblock.

```
FUNCTION GLOBAL __cdecl P_SSR_Malloc
VAR_INPUT
    size0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0      : PVOID;
END_VAR;
```

```
#define OS_SSR_Malloc(p1) _LSL_POS^.piSSR^.SSR_Malloc $ P_SSR_Malloc(p1)
```

Übergabeparameter		Typ	Beschreibung
size0		UDINT	Zu reservierende Byte
Rückgabeparameter		Typ	Beschreibung
ret0		PVOID	OS_SSR_Malloc liefert einen void Pointer auf den zugewiesenen Platz zurück oder, wenn nicht genügend Speicherkapazität zur Verfügung steht, wird NIL zurückgegeben. Um einen Pointer wieder auf einen anderen Typ als Void zurückzustellen, verwenden sie eine Type-Cast beim Rückgabewert.

Die OS_SSR_Malloc Funktion reserviert einen Speicherblock von mindestens size0 Bytes. Wegen des Platzbedarfs für die Abgleichungs- und Wartungsinformationen kann der Block größer werden als size0 Bytes.

2.3.3.2 OS_SSR_ReAlloc

Die OS_SSR_ReAlloc Funktion reserviert einen Speicherblock.

```
FUNCTION GLOBAL __cdecl P_SSR_ReAlloc
VAR_INPUT
    ptr0      : PVOID;
    size0     : UDINT;
END_VAR
VAR_OUTPUT
    ret0      : PVOID;
END_VAR;

#define OS_SSR_Realloc(p1,p2) _LSL_POS^.piSSR^.SSR_Realloc $ P_SSR_ReAlloc(p1,p2)
```

Übergabeparameter		Typ	Beschreibung
ptr0		PVOID	Zeiger auf einen vorher allokierten Speicherblock
size0		UDINT	Neue Größe in Byte
Rückgabeparameter		Typ	Beschreibung
ret0		PVOID	OS_SSR_ReAlloc liefert einen Void Pointer auf den umverteilten (und möglicherweise verschobenen) Speicherblock zurück. Der Rückgabewert ist NIL, wenn die Größe Null ist und das Puffer-Argument nicht gleich NIL ist, oder wenn nicht genügend Arbeitsspeicher verfügbar ist, um den Block auf die angegebene Größe zu erweitern. Im ersten Fall wird der ursprüngliche Block freigegeben. Im zweiten Fall bleibt der ursprüngliche Block unverändert. Für einen anderen Pointer als Void, verwenden sie eine Type-Cast beim Rückgabewert.

Die OS_SSR_ReAlloc Funktion ändert die Größe eines zugewiesenen Speicherblocks. Das ptr0 Argument zeigt auf den Anfang des Speicherbereichs. Ist ptr0 NIL, verhält sich S_SSR_ReAlloc genauso wie OS_SSR_Alloc und reserviert einen neuen Block von size0 Bytes. Ist ptr0 nicht NIL, sollte ein Pointer eines vorherigen Aufrufes von OS_SSR_Alloc oder OS_SSR_ReAlloc zurückgeliefert werden.

Das Argument size0 gibt die neue Größe des Blocks in Bytes an. Der Inhalt des Blocks bleibt bis zum Kürzesten der neuen und alten Größe unverändert, obwohl der neue Block an einer anderen Stelle sein kann.

Es ist nicht garantiert, dass es sich bei dem von OS_SSR_ReAlloc zurückgegebenen Pointer, um den durch das ptr0 Argument übertragenen handelt, da der neue Block an einem neuen Speicherort sein kann.

2.3.3.3 OS_SSR_Free

Die OS_SSR_Free Funktion gibt einen Speicherblock frei oder löscht einen.

```
FUNCTION GLOBAL __cdecl P_SSR_Free
VAR_INPUT
    ptr0      : PVOID;
END_VAR;
#define OS_SSR_Fr
ee(p1) _LSL_POS^.piSSR^.SSR_Free $ P_SSR_Free(p1)
```

Übergabeparameter	Typ	Beschreibung
ptr0	PVOID	Zugewiesener Speicherblock, der wieder freigegeben soll

2.4 API Serielle Schnittstelle

2.4.1 Übersicht

Die serielle Anwender-API bietet Interrupt-gesteuerte Kommunikation über serielle Schnittstellen. Empfangene Daten werden bis zu einer definierten Grenze gepuffert, bevor Daten verloren gehen. Für die Kommunikation über eine serielle Schnittstelle sind folgende Schritte notwendig:

- Öffnen Sie den Anschluss mit SERUSER_Init.
- Anwendungen, die Daten erhalten sollen, müssen SERUSER_RecvChar oder SERUSER_RecvBlock anrufen.
- Anwendungen, die Daten senden sollen, müssen SERUSER_Send anrufen.

- Ist die Kommunikation abgeschlossen, schließen Sie den Port mit SERUSER_Close und aktivieren Sie optional die LASAL-Online-Kommunikation wieder mit SERUSER_SetOnline.

2.4.2 Interface-Funktionen SERIAL

Header-Dateien: lsl_st_serial.h

2.4.2.1 SERUSER_ClearRecvBuffer

Die SERUSER_ClearRecvBuffer-Funktion löscht den Empfangspuffer.

```
FUNCTION GLOBAL __cdecl P_SerUsr_ClearRecvBuffer
VAR_INPUT
handle0      : pVoid;
END_VAR
VAR_OUTPUT
ret0        : DINT;
END_VAR;

#define SERUSER_ClearRecvBuffer(p1)
  _LSL_POS^.piSerial^.pClearRecvBuffer
  $ P_SerUsr_ClearRecvBuffer(p1)
```

Übergabeparameter	Typ	Beschreibung
handle0	pVoid	Handle für die serielle Schnittstelle, das von SERUSER_Init zurückgegeben wird
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	0 Funktion erfolgreich <0 Negativer Fehler-Code

2.4.2.2 SERUSER_Close

Die SERUSER_Close-Funktion schließt die serielle Schnittstelle.

```
FUNCTION GLOBAL __cdecl P_SerUsr_Close
VAR_INPUT
handle0      : pVoid;
END_VAR;

#define SERUSER_Close(p1)
  _LSL_POS^.piSerial^.pClose
  $ P_SerUsr_Close(p1)
```

Übergabeparameter	Typ	Beschreibung
handle0	pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird

Diese Funktion schaltet die LASAL Online-Kommunikation nicht wieder ein.

2.4.2.3 [SERUSER_EnableFIFO](#)

Die SERUSER_EnableFIFO-Funktion aktiviert oder deaktiviert den Betrieb des Empfangs- und Übergabe-FIFOs.

```
FUNCTION GLOBAL __cdecl P_SerUsr_EnableFIFO
VAR_INPUT
    handle0      : pVoid;
    Trigger0     : UDINT;
END_VAR
VAR_OUTPUT
    ret0         : DINT;
END_VAR;

#define SERUSER_EnableFIFO(p1,p2)
    _LSL_POS^.piSerial^.pEnableFIFO
    $ P_SerUsr_EnableFIFO(p1,p2)
```

Übergabeparameter	Typ	Beschreibung				
handle0	pVoid	Handle für die serielle Schnittstelle, das von SERUSER_Init() zurückgegeben wird.				
Trigger0	UDINT	Spezifiziert den Auslösungspegel des Empfangs-FIFOs. Wenn die Anzahl der Bytes im Empfangs-FIFO den Auslösungspegel erreicht, wird ein Empfangsdaten-Verfügbarkeits-Interrupt ausgelöst. Ein Wert von 0 deaktiviert den FIFO. Um das FIFO zu aktivieren, kann einer der folgenden Auslösungspegel angeben werden: 1, 4, 8 und 14.				
Rückgabeparameter	Typ	Beschreibung				
ret0	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: center;">0</td><td style="width: 90%;">Funktion erfolgreich</td></tr> <tr> <td style="text-align: center;"><0</td><td>Negativer Fehler-Code</td></tr> </table>	0	Funktion erfolgreich	<0	Negativer Fehler-Code
0	Funktion erfolgreich					
<0	Negativer Fehler-Code					

Wenn die Übertragungs- und Empfangs-FIFOs deaktiviert werden, gehen alle im FIFO-Puffer gespeicherten Daten verloren. Wenn diese Funktion aufgerufen wird, um die FIFO zu aktivieren, UART aber kein FIFO hat, dann schlägt die Funktion fehl.

Anforderungen

LasalOS: ab LasalOS 5.28 ist erforderlich.

2.4.2.4 SERUSER_Get422Mode

The SERUSER_Get422Mode() function queries the current selection of the RS422/485 output driver.

```
FUNCTION GLOBAL __cdecl P_SerUsr_Get422Mode
VAR_INPUT
    handle0      : pVoid;
    OnOff0       : ^UDINT;
END_VAR
VAR_OUTPUT
    ret0         : DINT;
END_VAR;

#define SERUSER_Get422Mode(p1,p2)
    _LSL_POS^.piSerial^.pGet422Mode
    $ P_SerUsr_Get422Mode(p1,p2)
```

Übergabeparameter		Typ	Beschreibung
handle0		pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init zurückgegeben wird
OnOff0		^UDINT	Eine Variable, die den aktuellen Wert des RS422/485-Output-Treibers erhält. Der Wert Null bedeutet, dass der Output-Treiber nicht ausgewählt ist.
Rückgabeparameter		Typ	Beschreibung
		DINT	0 Funktion erfolgreich <0 Negativer Fehler-Code

Anforderungen

LasalOS: ab LasalOS 5.28 ist erforderlich.

2.4.2.5 SERUSER_GetError

Die SERUSER_GetError-Funktion ruft den letzten Fehler-Code der seriellen Schnittstelle auf.

```
FUNCTION GLOBAL __cdecl P_SerUsr_GetError
VAR_INPUT
    handle0      : pVoid;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;

#define SERUSER_GetError(p1)
    _LSL_POS^.piSerial^.pGetError
    $ P_SerUsr_GetError(p1)
```

Übergabeparameter	Typ	Beschreibung
handle0	pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init zurückgegeben wird
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	Fehler-Code der letzten aufrufenden seriellen Schnittstelle. Der Rückgabewert kann einer der folgenden sein.
Code	Name	Beschreibung
0	SERERROR_NONE	Kein Fehler
-1	SERERROR_COMM	Entweder wurde eine ungültige Zahl der seriellen Schnittstelle spezifiziert oder die Funktion wird für diese Schnittstelle nicht unterstützt.
-2	SERERROR_BAUDTABLE	Fehler Baud-Tabelle.
-3	SERERROR_BAUDRATE	Der Baudrate-Parameter ist falsch.
-4	SERERROR_PARITY	Der Paritäts-Parameter ist falsch.
-5	SERERROR_STOPBIT	Der Stop-Bit-Parameter ist falsch.
-6	SERERROR_WORDLEN	Der wordlen-Parameter ist falsch.
-10	SERERROR_INUSE	Die angegebene serielle Schnittstelle wird bereits von der Anwendung verwendet.

-11	SERERROR_OSINUSE	COM-Schnittstelle wird bereits vom OS verwendet.
-12	SERERROR_NOTAVAIL	Die angegebene serielle Schnittstelle ist auf diesem System nicht verfügbar.
-13	SERERROR_NOMEM	Es ist nicht genügend Speicherplatz verfügbar, um diesen Befehl zu bearbeiten.
-14	SERERROR_NOHANDLE	Das Handle ist ungültig.
-15	SERERROR_PARAMETER	Der Parameter ist falsch.
-16	SERERROR_RECVBUF	Die Empfangspuffer-Parameter sind falsch.
-17	SERERROR_SENDBUF	Die Sendepuffer-Parameter sind falsch.
-19	SERERROR_RECVERROR	Die Funktionen SERUSER_RecvChar oder SERUSER_RecvBlock wurden aufgerufen, aber es befinden sich keine Daten im Empfangspuffer.
-20	SERERROR_SENDERROR	Der Sendevorgang konnte nicht abgeschlossen werden (Unterbrechung, Sendepuffer voll).
-21	SERERROR_GENERAL	Interner Fehler beim Initialisieren oder bei der Bedienung der seriellen Schnittstelle.
-22	SERERROR_PARITY_E	Empfangene Datenpakete wurden aufgrund einer ungültigen Parität verworfen.
-23	SERERROR_FRAMING_E	Empfangene Datenpakete wurden aufgrund von Framing-Fehlern (z.B. falsche Baudrate, Wortlänge usw.) verworfen.

2.4.2.6 SERUSER_GetInfo

Die SERUSER_GetInfo()-Funktion ruft die aktuellen Ansteuerungs-Einstellungen und verwendeten Ressourcen für eine bestimmte serielle Schnittstelle auf.

```
FUNCTION GLOBAL __cdecl P_SerUsr_GetInfo
VAR_INPUT
    handle0      : pVoid;
    info0        : ^LSLAPI_SERIALINFO;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;

#define SERUSER_GetInfo(p1,p2)
    _LSL_POS^.piSerial^.pGetInfo
```

```
$ P_SerUsr_GetInfo(p1, p2)
```

Übergabeparameter		Typ	Beschreibung	
handle0		pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird	
info0		^LSLAPI_SERIALINFO	Zeiger auf eine LSLAPI_SERIALINFO-Struktur, die Informationen erhält	
Rückgabeparameter		Typ	Beschreibung	
ret0		DINT	0 Funktion erfolgreich <0 Fehler	

LSLAPI_SERIALINFO Struktur

```
LSLAPI_SERIALINFO : STRUCT
  initialized      : USINT;
  comportnum       : USINT;
  IRQNum          : USINT;
  IOPort           : UINT;
  Baudrate         : UINT;
  Ptr_RecvBuffer   : pVoid;
  Ptr_SendBuffer   : pVoid;
END_STRUCT;
```

Elemente der LSLAPI_SERIALINFO-Struktur.

Initialized	Gibt an, ob die serielle Schnittstelle initialisiert ist
Comportnum	0-basierter Index der seriellen Schnittstelle  Zu beachten ist, dass Common-Parameter in SERUSER-Funktionen 1-basierend sind!
IRQNum	IRQ-Nummer der seriellen Schnittstelle
IOPort	I/O-Basisadresse der seriellen Schnittstelle
Baudrate	Baudrate, mit der die serielle Schnittstelle arbeitet
Ptr_RecvBuffer	Zeiger auf den Empfangspuffer
Ptr_SendBuffer	Zeiger auf den Sendepuffer

Diese Funktion sollte nur zu Informationszwecken verwendet werden. Es wird z.B. nicht empfohlen, auf die Register der seriellen Schnittstelle direkt zuzugreifen!

2.4.2.7 SERUSER_GetModemControl

Die SERUSER_GetModemControl()-Funktion fragt das Modem-Control-Register (MCR) ab.

```
FUNCTION GLOBAL __cdecl P_SerUsr_GetModemControl
VAR_INPUT
    handle0      : pVoid;
    Value0       : ^USINT;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;

#define SERUSER_GetModemControl(p1,p2)
    _LSL_POS^.piSerial^.pGetModemControl
    $ P_SerUsr_GetModemControl(p1,p2)
```

Übergabeparameter	Typ	Beschreibung
handle0	pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird
Value0	^USINT	Zeiger auf ein USINT, das den Inhalt des Modem-Control-Registers (MCR) erhält
Rückgabeparameter	Typ	Beschreibung
	DINT	0 Funktion erfolgreich <0 Negativer Fehler-Code

Anforderungen

LasalOS: LasalOS ab 5.28 wird benötigt

2.4.2.8 SERUSER_GetModemStatus

Die SERUSER_GetModemStatus-Funktion fragt das Modem-Status-Register (MSR) ab.

```
FUNCTION GLOBAL __cdecl P_SerUsr_GetModemStatus
VAR_INPUT
    handle0      : pVoid;
    Value0       : ^USINT;
END_VAR
```

```

VAR_OUTPUT
    ret0          : DINT;
END_VAR;

#define SERUSER_GetModemStatus(p1,p2)
    _LSL_POS^.piSerial^.pGetModemStatus
    $ P_SerUsr_GetModemStatus(p1,p2)

```

Übergabeparameter	Typ	Beschreibung	
handle0	pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird	
Value0	^USINT	Zeiger auf ein USINT, das den Inhalt des Modem-Status-Registers (MSR) erhält	
Rückgabeparameter	Typ	Beschreibung	
ret0	DINT	0 Funktion erfolgreich <0 Negativer Fehler-Code	

Anforderungen

LasalOS: LasalOS ab 5.28 wird benötigt

2.4.2.9 SERUSER_GetRecvStatus

Die SERUSER_GetRecvStatus()-Funktion liefert die Anzahl der Zeichen im Empfangspuffer.

```

FUNCTION GLOBAL __cdecl P_SerUsr_GetRecvStatus
VAR_INPUT
    handle0 : pVoid;
END_VAR
VAR_OUTPUT
    ret0 : DINT;
END_VAR;

#define SERUSER_GetRecvStatus(p1)
    _LSL_POS^.piSerial^.pGetRecvStatus
    $ P_SerUsr_GetRecvStatus(p1)

```

Übergabeparameter	Typ	Beschreibung	
handle0	pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird	
Rückgabeparameter	Typ	Beschreibung	

ret0	DINT	>0 Anzahl der Zeichen derzeit im Empfangspuffer <0 Negativer Fehler-Code
------	------	-----------------------------------------------------------------------------

2.4.2.10 SERUSER_GetSendStatus

Die SERUSER_GetSendStatus()-Funktion die Anzahl der Zeichen im Sendepuffer zurück.

```
FUNCTION GLOBAL __cdecl P_SerUsr_GetSendStatus
VAR_INPUT
    handle0 : pVoid;
END_VAR
VAR_OUTPUT
    ret0 : DINT;
END_VAR;

#define SERUSER_GetSendStatus(p1)
    _LSL_POS^.piSerial^.pGetSendStatus
    $ P_SerUsr_GetSendStatus(p1)
```

Übergabeparameter	Typ	Beschreibung
handle0	pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird
Rückgabeparameter	Typ	Beschreibung
ret0	DINT	>0 Anzahl der Zeichen im Sendepuffer <0 Negativer Fehler-Code

2.4.2.11 SERUSER_Init

Die SERUSER_Init()-Funktion öffnet eine serielle Schnittstelle, konfiguriert diese und liefert ein Handle zurück, das benutzt werden kann, um auf die Schnittstelle zuzugreifen. Wird die Schnittstelle auch von der LASAL Online-Kommunikation verwendet, wird die LASAL Online-Kommunikation deaktiviert. Die LASAL Online-Kommunikation wird wieder aktiviert, wenn das Projekt gestoppt wird, oder wenn es einen Aufruf von [SERUSER_SetOnline\(\)](#) ermöglicht.

```
FUNCTION GLOBAL __cdecl P_SerUsr_Init
VAR_INPUT
    comnum0 : UINT;
    combaud0 : UINT;
    parity0 : UINT;
    stopbits0 : UINT;
    wordlength0 : UINT;
```

```

END_VAR
VAR_OUTPUT
    ret0      : pVoid;
END_VAR;

#define SERUSER_Init(p1,p2,p3,p4,p5)
    _LSL_POS^.piSerial^.pInitSerial
    $ P_SerUsr_Init(p1,p2,p3,p4,p5)

```

Übergabeparameter	Typ	Beschreibung																						
comnum0	UINT	<p>Gibt die Nummer der seriellen Schnittstelle an</p> <table> <tr><td>1</td><td>SERUSERCOM_1</td></tr> <tr><td>2</td><td>SERUSERCOM_2</td></tr> <tr><td>3</td><td>SERUSERCOM_3</td></tr> <tr><td>4</td><td>SERUSERCOM_4</td></tr> </table>	1	SERUSERCOM_1	2	SERUSERCOM_2	3	SERUSERCOM_3	4	SERUSERCOM_4														
1	SERUSERCOM_1																							
2	SERUSERCOM_2																							
3	SERUSERCOM_3																							
4	SERUSERCOM_4																							
combaud0	UINT	<p>Gibt die Baud-Rate an, bei der die serielle Schnittstelle betrieben wird</p> <table> <tr><td>0</td><td>SERUSERBAUD_300</td></tr> <tr><td>1</td><td>SERUSERBAUD_600</td></tr> <tr><td>2</td><td>SERUSERBAUD_1200</td></tr> <tr><td>3</td><td>SERUSERBAUD_2400</td></tr> <tr><td>4</td><td>SERUSERBAUD_4800</td></tr> <tr><td>5</td><td>SERUSERBAUD_9600</td></tr> <tr><td>6</td><td>SERUSERBAUD_14400</td></tr> <tr><td>7</td><td>SERUSERBAUD_19200</td></tr> <tr><td>8</td><td>SERUSERBAUD_38400</td></tr> <tr><td>9</td><td>SERUSERBAUD_56000</td></tr> <tr><td>10</td><td>SERUSERBAUD_115200</td></tr> </table>	0	SERUSERBAUD_300	1	SERUSERBAUD_600	2	SERUSERBAUD_1200	3	SERUSERBAUD_2400	4	SERUSERBAUD_4800	5	SERUSERBAUD_9600	6	SERUSERBAUD_14400	7	SERUSERBAUD_19200	8	SERUSERBAUD_38400	9	SERUSERBAUD_56000	10	SERUSERBAUD_115200
0	SERUSERBAUD_300																							
1	SERUSERBAUD_600																							
2	SERUSERBAUD_1200																							
3	SERUSERBAUD_2400																							
4	SERUSERBAUD_4800																							
5	SERUSERBAUD_9600																							
6	SERUSERBAUD_14400																							
7	SERUSERBAUD_19200																							
8	SERUSERBAUD_38400																							
9	SERUSERBAUD_56000																							
10	SERUSERBAUD_115200																							
parity0	UINT	<p>Gibt die Paritätsregelung an, die verwendet werden soll.</p> <table> <tr><td>0</td><td>SERUSERPARITY_NONE</td></tr> <tr><td>1</td><td>SERUSERPARITY_ODD</td></tr> <tr><td>2</td><td>SERUSERPARITY_EVEN</td></tr> </table>	0	SERUSERPARITY_NONE	1	SERUSERPARITY_ODD	2	SERUSERPARITY_EVEN																
0	SERUSERPARITY_NONE																							
1	SERUSERPARITY_ODD																							
2	SERUSERPARITY_EVEN																							

		3 SERUSERPARITY_MARK 4 SERUSERPARITY_SPACE
stopbits0 :	UINT	Gibt die Anzahl der Stop-Bits an, die verwendet werden sollen 1 1 Stop-Bit 2 2 Stop-Bits für Wörter mit einer Länge von 6, 7 oder 8 Bits oder 1,5 Stop-Bits für Wortlängen von 5 Bits.
wordlength0	UINT	Gibt die Anzahl der Bits in den übertragenen und empfangenen Byte an. Mögliche Werte sind 5, 6, 7, 8. Eine Wortlänge von 8 Bits wird am häufigsten verwendet.
Rückgabeparameter	Typ	Beschreibung
ret0	pVoid	Ist die Funktion erfolgreich, ist der Rückgabewert ein offenes Handle zur angegebenen seriellen Schnittstelle. Ist die Funktion fehlschlagen, ist der Rückgabewert ein NIL-Zeiger. Um erweiterte Fehlerinformationen abzufragen, rufen Sie SERUSER_GetError() auf.

Verwenden Sie [SERUSER_Close\(\)](#) zum Schließen der seriellen Schnittstelle.

2.4.2.12 SERUSER_RecvBlock

Die SERUSER_RecvBlock()-Funktion liest Daten aus dem Sendepuffer des seriellen Treibers.

```
FUNCTION GLOBAL __cdecl P_SerUsr_RecvBlock
VAR_INPUT
  handle0      : pVoid;
  buffer0      : pVoid;
  rdlength0    : UDINT;
  rdlen0       : ^UDINT;
END_VAR
VAR_OUTPUT
  ret0         : DINT;
END_VAR;

#define SERUSER_RecvBlock(p1,p2,p3,p4)
  _LSL_POS^.piSerial^.pRecvBlock
  $ P_SerUsr_RecvBlock(p1,p2,p3,p4)
```

Übergabeparameter	Typ	Beschreibung
handle0	pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird

buffer0	pVoid	Zeiger auf den Puffer, der die Daten aus dem Empfangspuffer liest				
rdlength0	UDINT	Gibt die Anzahl der zu lesenden Byte an				
rlen0	^UDINT	Zeiger auf ein UDINT, um die Anzahl der tatsächlich gelesenen Bytes zu erhalten. Dieser Wert kann kleiner als rdlength0 sein.				
Rückgabeparameter	Typ	Beschreibung				
ret0	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: right;">0</td><td>Erfolg, mindestens ein Byte wurde verfügbar</td></tr> <tr> <td style="text-align: right;"><0</td><td>Negativer Fehler-Code</td></tr> </table>	0	Erfolg, mindestens ein Byte wurde verfügbar	<0	Negativer Fehler-Code
0	Erfolg, mindestens ein Byte wurde verfügbar					
<0	Negativer Fehler-Code					

2.4.2.13 SERUSER_RecvChar

Die SERUSER_RecvChar()-Funktion liest ein Byte vom Empfangspuffer des seriellen Treibers

```
FUNCTION GLOBAL __cdecl P_SerUsr_RecvChar
VAR_INPUT
    handle0 : pVoid;
    buffer0 : pVoid;
END_VAR
VAR_OUTPUT
    ret0 : DINT;
END_VAR;

#define SERUSER_RecvChar(p1,p2)
    _LSL_POS^.piSerial^.pRecvChar
    $ P_SerUsr_RecvChar(p1,p2)
```

Übergabeparameter	Typ	Beschreibung				
handle0	pVoid	Handle für die serielle Schnittstelle, das von SERUSER_Init() zurückgegeben wird				
buffer0	pVoid	Zeiger auf ein Zeichen, das das Byte aus dem Empfangspuffer erhält				
Rückgabeparameter	Typ	Beschreibung				
ret0	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: right;">0</td><td>Erfolg, ein Byte wurde verfügbar</td></tr> <tr> <td style="text-align: right;"><0</td><td>Negativer Fehler-Code</td></tr> </table>	0	Erfolg, ein Byte wurde verfügbar	<0	Negativer Fehler-Code
0	Erfolg, ein Byte wurde verfügbar					
<0	Negativer Fehler-Code					

2.4.2.14 SERUSER_Send

SERUSER_Send() überträgt Daten asynchron unter der Verwendung von Interrupts. Die Daten werden in einen Puffer gespeichert und vom Interrupt-Handler übertragen, sobald das Übertragungsregister leer wird.

```
FUNCTION GLOBAL __cdecl P_SerUsr_Send
VAR_INPUT
handle0      : pVoid;
buffer0      : pVoid;
bufferlength0 : UDINT;
wrlen0       : ^UDINT;
END_VAR
VAR_OUTPUT
ret0         : DINT;
END_VAR;

#define SERUSER_Send(p1,p2,p3,p4)
    _LSS_POS^.piSerial^.pSend
    $ P_SerUsr_Send(p1,p2,p3,p4)
```

Übergabeparameter	Typ	Beschreibung						
handle0	pVoid	Handle für die serielle Schnittstelle, das von SERUSER_Init() zurückgegeben wird						
buffer0	pVoid	Zeiger auf einen Puffer, der die zu übertragenden Daten enthält						
bufferlength0	UDINT	Länge des zu übertragenden Datenpuffers						
wrlen0	^UDINT	Zeiger auf ein UDINT, um die Anzahl der Byte im Sendepuffer zu erhalten. wrlen0^ enthält nach dem Aufruf normalerweise bufferlength0. Im Fall eines Sendepuffer-Mangels könnte der zurückgegebene Wert vielleicht weniger sein. wrlen0 kann auf NIL gesetzt werden, wenn diese Information von einer Anwendung nicht erforderlich ist.						
Rückgabeparameter	Typ	Beschreibung						
ret0	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td><td style="width: 10%;">0</td><td>Funktion erfolgreich</td></tr> <tr> <td></td><td><0</td><td>Fehler</td></tr> </table>		0	Funktion erfolgreich		<0	Fehler
	0	Funktion erfolgreich						
	<0	Fehler						

2.4.2.15 SERUSER_Set422Mode

Die SERUSER_Set422Mode()-Funktion steuert den Ouput-Treiber auf eine RS422/RS485-Schnittstelle.

```
FUNCTION GLOBAL __cdecl P_SerUsr_Set422Mode
VAR_INPUT
    handle0      : pVoid;
```

```

OnOff0      : UDINT;
END_VAR
VAR_OUTPUT
  ret0      : DINT;
END_VAR;

#define SERUSER_Set422Mode(p1,p2)
  _LSL_POS^.piSerial^.pSet422Mode
  $ P_SerUsr_Set422Mode(p1,p2)

```

Übergabeparameter	Typ	Beschreibung	
handle0	pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird	
recvbuffer0	UDINT	0	Ausgangstreiber ist ausgeschaltet
		1	Ausgangstreiber ist eingeschaltet
Übergabeparameter	Typ	Beschreibung	
ret0	DINT	0	Funktion erfolgreich
		<0	Fehler

Der RS422- und RS485-Modus wird nicht von allen Plattformen unterstützt. Bitte sehen Sie in Ihrer Hardware-Dokumentation nach. Da RS422 eine Punkt-zu-Punkt-Verbindung ist, kann der Ausgabe-Treiber in diesem Modus immer eingeschaltet sein. Da RS485 ein Multimaster-Verbindungstyp ist, kann der Output-Treiber im RS485 nur eingeschaltet sein, wenn Daten gesendet werden.

Anforderungen

LasalOS: LasalOS ab 5.28 wird benötigt

2.4.2.16 SERUSER_SetBufferRecv

Diese Funktion spezifiziert einen neuen Empfangs-Puffer

```

FUNCTION GLOBAL __cdecl P_SerUsr_SetBufferRecv
VAR_INPUT
  handle0      : pVoid;
  recvbuffer0  : pVoid;
  bufferlength0 : UDINT;
END_VAR
VAR_OUTPUT
  ret0      : DINT;
END_VAR;

```

```
#define SERUSER_SetBufferRecv(p1,p2,p3)
  _L5L_POS^.piSerial^.pSetBufferRecv
  $ P_SerUsr_SetBufferRecv(p1,p2,p3)
```

Übergabeparameter		Typ	Beschreibung	
handle0		pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird	
recvbuffer0		pVoid	Zeiger auf eine Empfangspuffer	
bufferlength0		UDINT	Größe von recvbuffer0 in Byte; dieser Wert muss mindestens 128 Byte sein	
Rückgabeparameter		Typ	Beschreibung	
ret0		DINT	0	Funktion erfolgreich
			<0	Fehler

2.4.2.17 SERUSER_SetModemControl

Die SERUSER_SetModemControl()-Funktion setzt oder löscht einzelne Bits des Modem-Control-Registers (MCR) des UARTs. Für eine genauere Beschreibung dieses Registers lesen Sie bitte in der UART-Dokumentation nach.

```
FUNCTION GLOBAL __cdecl P_SerUsr_SetModemControl
VAR_INPUT
  handle0      : pVoid;
  SetToOneZero0 : UDINT;
  NewValue0    : UDINT;
END_VAR
VAR_OUTPUT
  ret0        : DINT;
END_VAR;

#define SERUSER_SetModemControl(p1,p2,p3)
  _L5L_POS^.piSerial^.pSetModemControl
  $ P_SerUsr_SetModemControl(p1,p2,p3)
```

Übergabeparameter		Typ	Beschreibung	
handle0		pVoid	Handle für die serielle Schnittstelle, der von SERUSER_Init() zurückgegeben wird	
SetToOneZero0		UDINT	Dieser Parameter legt fest, welche Operation im Modem-Control-Register ausgeführt werden soll. Ein Wert von 0 führt eine bitweise logische OR-Operation mit MCR und NewValue0 aus. Ein Wert von 1 führt eine bitweise logische NOT-Operation mit NewValue0 aus und dann eine bitweise logische AND-Operation mit MCR mit den	

		negierten NewValue0. Das Ergebnis dieser Operation wird auf das Modem-Control-Register geschrieben. Mit SetToOneZero0 = 0 können einzelne Bits gesetzt werden und mit SetToOneZero0 = 1 können einzelne Bits des MCR gelöscht werden.				
NewValue0	UDINT	Enthält den Wert, der zur Berechnung des neuen Wertes des MCRs verwendet wird. Bitte sehen Sie in der Beschreibung des Parameters SetToOneZero0 nach.				
Rückgabeparameter	Typ	Beschreibung				
ret0	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td><td>Funktion erfolgreich</td></tr> <tr> <td><0</td><td>Negativer Fehler-Code</td></tr> </table>	0	Funktion erfolgreich	<0	Negativer Fehler-Code
0	Funktion erfolgreich					
<0	Negativer Fehler-Code					

Anforderungen

LasalOS: LasalOS ab 5.28 wird benötigt

2.4.2.18 SERUSER_SetOnline

Die SERUSER_SetOnline()-Funktion aktiviert oder deaktiviert die LASAL Online-Kommunikation. Wurde die LASAL Online-Kommunikation mit der [SERUSER_Init\(\)](#)-Funktion deaktiviert, verwenden sie diese Funktion, um die LASAL Online-Kommunikation wieder zu aktivieren.

```
FUNCTION GLOBAL __cdecl P_SerUsr_SetOnline
VAR_INPUT
    action0 : UDINT;
END_VAR
VAR_OUTPUT
    ret0 : DINT;
END_VAR;

#define SERUSER_SetOnline(pl)
    _LSL_POS^.piSerial^.pSetOnline
    $ P_SerUsr_SetOnline(pl)
```

Übergabeparameter	Typ	Beschreibung				
action0	UDINT	Gibt an, ob die LASAL Online-Kommunikation aktiviert werden soll				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td><td>Ausgeschaltet</td></tr> <tr> <td>1</td><td>Aktiviert</td></tr> </table>	0	Ausgeschaltet	1	Aktiviert
0	Ausgeschaltet					
1	Aktiviert					

Rückgabeparameter	Typ	Beschreibung
ret0	DINT	0 Funktion erfolgreich <0 Fehler



Diese Funktion darf nicht aufgerufen werden, wenn die serielle Schnittstelle offen ist.

2.4.2.19 SERUSER_UserFunction

Die SERUSER_UserFunction()-Funktion dient zur Installation einer High-Level-Benutzer-Interrupt-Handler-Funktion. Wird ein Interrupt über die serielle Schnittstelle ausgelöst, ruft der Low-Level-Handler im Betriebssystem den High-Level-Handler auf. Das Betriebssystem löscht den Interrupt nicht vom UART. Die Anwenderfunktion ist für die Ermittlung des Grundes des Interrupts und das Lesen des entsprechenden Registers, um den Interrupt vom UART zu löschen, verantwortlich.

```

FUNCTION GLOBAL __cdecl P_SerUsr_SetFunction
VAR_INPUT
  handle0  : pVoid;
  function0 : pVoid;
  param0   : pVoid;
END_VAR
VAR_OUTPUT
  ret0      : DINT;
END_VAR;

#define SERUSER_UserFunction(p1,p2,p3)
  _LSL_POS^.piSerial^.pSetFunction
  $ P_SerUsr_SetFunction(p1,p2,p3)

```

Übergabeparameter	Typ	Beschreibung
handle0	pVoid	Handle für die serielle Schnittstelle, das von SERUSER_Init zurückgegeben wird
function0	pVoid	Zeiger auf die User Interrupt-Funktion. Dieser Funktionstyp sieht wie folgt aus: <pre> FUNCTION GLOBAL __cdecl P_SerUsr_UsrIntFcn VAR_INPUT userParam : pVoid; comNum : pVoid; ioPort : pVoid; END_VAR </pre>

UserParam	pVoid	Zeiger, der im Parameter param0 dieser Funktion angegeben wurde				
comNum	pVoid	Es handelt sich um einen 0-basierten Index der seriellen Schnittstelle  Zu beachten ist, dass Common-Parameter in SERUSER-Funktionen 1-basierend sind!				
ioPort	pVoid	I/O-Basisadresse der seriellen Schnittstelle				
param0	pVoid	Enthält einen Zeiger zur Übergabe an die Anwender Interrupt-Funktion				
Rückgabeparameter	Typ	Beschreibung				
ret0	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">0</td><td>Erfolg</td></tr> <tr> <td style="text-align: center;"><0</td><td>Fehler</td></tr> </table>	0	Erfolg	<0	Fehler
0	Erfolg					
<0	Fehler					

Anforderungen

LasalOS: LasalOS ab 5.28 wird benötigt

2.5 API Seriennummer

2.5.1 Allgemein

Diese Schnittstelle stellt die seriellen SPS- und IDE-Laufwerksnummern zur Verfügung.

2.5.2 Strukturen, die in ISYSSENRNUM verwendet werden

Header files: Isl_st_syssernum.h

2.5.2.1 tagPLCInfo

Struktur, die die SPS-Info enthält.

```
// PLC Info
typedef struct tagPLCInfo
{
    // Version
```

```
unsigned long ulVersion;

// BIOS Version
char szBIOSVersion[ 16 ];

// Serial number
char szSerialNumber[ 24 ];

// Application
char szApplication[ 128 ];

} PLCINFO, *PPLCINFO;

// PLC Info
TYPE
LSL_PLCINFO : STRUCT

//
// Version
//
udVersion : UDINT;

// BIOS Version
szBIOSVersion : ARRAY [0..15] OF CHAR;

// Serial number
szSerialNumber : ARRAY [0..23] OF CHAR;

// Application
szApplication : ARRAY [0..127] OF CHAR;

END_STRUCT;
END_TYPE
```

Parameter

udVersion:	Versionsnummer dieser Struktur. Zukünftige Versionen können mehr Komponenten haben.
szBIOSVersion	BIOS-Versionsnummer der SPS, falls vorhanden
szSerialVersion	Seriennummer der SPS
szApplication	Name des aktuellen Programms

Diese Komponenten sind zurzeit in IPCs und C-IPCs verfügbar.

Anforderungen

Version: Ab LasalOS 5.51.

2.5.3 Interface-Funktionen ISYSSERNUM

Header-Dateien: lsl_st_syssernum.h

Dieser Abschnitt beschreibt die Schnittstelle für die Seriennummern. Um diese Schnittstelle zu verwenden, muss zuerst ein Zeiger über OS_CILGET Daten erhalten. Der Name der Schnittstelle ist ISYSSERNUM. Wird wie im folgenden Beispiel der gleiche Name für den Zeiger benutzt, können die vordefinierten Makros zur Verringerung und Vereinfachung der Programmierung verwendet werden.

Die Seriennummern werden als nicht-Null begrenzte ASCII-Zeichenketten zurückgegeben!

2.5.3.1 SernumGetPLC

Ruft die Seriennummer der SPS ab.

```
int SernumGetPLC(
    unsigned char *pSerNum,
    unsigned long ulBufLen
);
FUNCTION __CDECL GLOBAL P_SernumGetPLC
VAR_INPUT
    pSerNum      : ^Void;
    ulBufLen    : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
#define ISYSSERNUM_SERNUMGETPLC(p1,p2)
    pISysSernum^.SernumGetPLC
    $ P_SernumGetPLC(p1,p2)
```

Übergabeparameter		Typ	Beschreibung	
perNum		^Void	Zeiger auf den Puffer, der die Seriennummer empfangen wird	
ulBufLen		UDINT	Größe des Puffers	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	≥ 0 Anzahl der Zeichen, die in den Puffer kopiert wurden <0 Fehler	

Beispiel

```
#include " lsl_st_syssernum.h"
```

```

VAR_GLOBAL
  PISysSernum  : ^LSL_ISYSSERNUM;
  Drive        : USINT;
  Buffer       : array [0..19] of USINT;
  Erg          : DINT;
END_VAR

FUNCTION NewClass0::NewClass0
VAR_OUTPUT
  ret_code      : CONFSTATES;
END_VAR

  if OS_CILGET("ISYSSERNUM", #pISysSernum) then
  else
    Erg := ISYSSERNUM_SERNUMGETPLC( #Buffer, 20 );
    Drive := 'c';
    Erg := ISYSSERNUM_SERNUMGETPLCDRIVE( Drive, #Buffer, 20 );
  end_if
  ret_code := C_OK;
END_FUNCTION // NewClass0::NewClass0

```

Anforderungen

Version: Ab LasalOS 5.43

2.5.3.2 SernumGetPLCDrive

Ruft die Seriennummer der IDE- und USB-Laufwerke der SPS ab, falls vorhanden.

```

int SernumGetPLCDrive(
unsigned char ucDrive,
unsigned char *pSerNum,
unsigned long ulBufLen
);
FUNCTION __CDECL GLOBAL P_SernumGetPLCDrive
VAR_INPUT
  ucDrive   : USINT;
  pSerNum   : ^Void;
  ulBufLen  : UDINT;
END_VAR
VAR_OUTPUT
  retval   : DINT;
END_VAR;
#define ISYSSERNUM_SERNUMGETPLCDRIVE(p1,p2,p3)
  pISysSernum^.SernumGetPLCDrive
  $ P_SernumGetPLCDrive(p1,p2,p3)

```

Übergabeparameter	Typ	Beschreibung
ucDrive	USINT	Laufwerksbuchstabe des gewünschte Laufwerks; der verfügbare Bereich ist von A bis Z und a bis z gültig

pSerNum	^Void	Zeiger auf den Puffer, der die Seriennummer empfangen wird
ulBufLen	UDINT	Größe des Puffers
Rückgabeparameter	Typ	Beschreibung
retval	DINT	<div style="display: flex; justify-content: space-between; align-items: center;"> ≥0 Anzahl der Zeichen, die in den Puffer kopiert wurden </div> <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 5px;"> <0 Fehler </div>

Beispiel

Siehe oben

Anforderungen

Version: Ab LasalOS 5.43

2.5.3.3 SernumGetPLCInfo

Ruft einige allgemeine Informationen der SPS ab.

```
PPLCINFO SernumGetPLCInfo(
void
);
FUNCTION __CDECL GLOBAL P_SernumGetPLCInfo
VAR_INPUT
END_VAR
VAR_OUTPUT
    retval : ^LSL_PLCINFO;
END_VAR;
#define ISYSSERNUM_SERNUMGETPLCINFO()
    pISysSernum^.SernumGetPLCInfo
    $ P_SernumGetPLCInfo()
```

Übergabeparameter	Typ	Beschreibung
keine		
Rückgabeparameter	Typ	Beschreibung
retval	^LSL_PLCINFO	Zeiger auf eine PLC INFO-Struktur

Beispiel

Siehe oben

Anforderungen

Version: Ab LasalOS 5.51

2.6 API Sysmsg-Schnittstelle

2.6.1 Übersicht

Die sysmsg API-Funktionen erlauben der Anwendung Fehler, Warnungen und anderen Nachrichten auszugeben. Es ist wichtig, dass diese Nachrichten in eine Datei geschrieben werden können, um sie später nachlesen zu können.

Eine Meldung wird nicht sofort bei der Erstellung in eine Datei geschrieben, stattdessen wird diese zuerst in einen Puffer geschrieben. Dieser Puffer wird in eine Datei geschrieben, wenn die Anwendung beendet oder zurückgesetzt wird, eine Ausnahme aufgetreten ist oder ein Ausschalten erkannt wird. Zusätzlich können die Nachrichten durch den Aufruf einer bestimmten sysmsg API-Funktion in eine Datei geschrieben werden.

Da die Größe des Nachrichten-Puffers begrenzt ist können Meldungen verloren gehen, und zwar, wenn der Puffer voll ist, bevor dessen Inhalt in eine Datei geschrieben wird. Es gibt kein Tool mit dem der Puffer automatisch in eine Datei geschrieben werden kann, da das ständige Schreiben von Daten auf eine Festplatte gefährlich ist und die Lebensdauer der Festplatte verringert. Es ist wichtig, dass der Anwendungsprogrammierer dies beim Schreiben von Nachrichten in eine Datei beachtet. Im Falle eines Puffer-Overflows zeichnet das Betriebssystem die Zeit genau auf, wann der Overflow aufgetreten ist. Ein zusätzlicher Eintrag, dass ein Overflow aufgetreten ist, wird generiert, wenn der Puffer das nächste Mal in eine Datei geschrieben wird.

Eine Begrenzung, Dateiquote genannt, kann für die Nachrichtendatei definiert werden, um das Vollschreiben der Festplatte zu vermeiden. Wenn die Nachrichtengröße die Dateiquote überschreitet, wird die Nachricht in eine Backup-Datei umbenannt und eine neue Nachrichtendatei erstellt. Ist dies der Fall, geht eine bereits vorhandene Backup-Datei verloren. Daher ist die erforderliche Größe der Festplatte für Nachrichtendateien $2 * \text{Dateiquote}$; der Dateiname ist EVENTxx.LOG, wobei xx eine eindeutige Nummer für jedes Nachrichtenpuffer-Objekt darstellt. Der Pfad wo die Nachricht gespeichert wird, kann mit der Umgebungsvariablen APPMSGPATH eingestellt werden. Der Standardpfad ist C:\SYSMSG.

Sind nur die letzten Nachrichten wichtig, kann ein Puffer-Overflow akzeptiert werden. Die Anwendung schreibt den Inhalt des Nachrichten-Puffers nicht in eine Datei. Dies erfolgt beim Beenden der Anwendung durch das Betriebssystem. Ansonsten muss die

Anwendung den im Nachrichtenpuffer verwendeten Platz überwachen und eventuell den Inhalt des Puffers in eine Datei schreiben.

Um Nachrichten aus einer Anwendung zu generieren, müssen folgende Schritte durchgeführt werden:

- Erstellen Sie ein Nachrichtenpuffer-Objekt mit OS_SYSMSG_LCREATE.
- Wenn das Nachrichtenpuffer-Objekt in verschiedenen Teilen des Anwendungsprogramms verwendet werden soll, kann zusätzliches Handeln für das zuvor erstellte Objekt mit einem Aufruf von OS_SYSMSG_LOPEN angefordert werden.
- Schreiben von Daten oder Text auf den Nachrichtenpuffer entweder mit der OS_SYSMSG_LWRITEx, OS_SYSMSG_LPRINTFLNx oder OS_SYSMSG_LWRITE_Ix Funktion; x ist ein Platzhalter für verschiedene Ausdrucksformen dieser Funktionen.
- Die Größe des belegten Speicherplatzes im Puffer kann ebenfalls mit OS_SYSMSG_LINFO überwacht und der Inhalt mit OS_SYSMSG_LFLUSH in eine Datei geschrieben werden.
- Die OS_SYSMSG_LCLOSE Funktion kann verwendet werden, um Nachrichtenpuffer Handles zu schließen, wenn die Objekte nicht mehr benötigt werden. Jedoch geschieht dies durch das Betriebssystem beim Beenden der Anwendung.

Die sysmsg API-Funktionen sind thread-save, d.h. sie können in jedem Task aufgerufen werden und müssen nicht nacheinander sequenziell sein. Außer OS_SYSMSG_LWRITE_I0-4, diese Funktionen können nicht von einem Interrupt aufgerufen werden.

2.6.2 Interface-Funktionen SYSMSG

Header Dateien:isl_st_sysmsg.h

2.6.2.1 OS_SYSMSG_LCLOSE

Die OS_SYSMSG_LCLOSE()-Funktion schließt ein offenes Handle eines Nachrichtenpuffer-Objekts.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LCLOSE
VAR_INPUT
    hLog      : UDINT;
END_VAR;

#define OS_SYSMSG_LCLOSE(p1)
    OS_pLslSysMsg^.lclose
    $ P_LSLSYSMSG_LCLOSE(p1)
```

Übergabeparameter	Typ	Beschreibung
hLog	UDINT	Handle auf das Nachrichtenpuffer-Objekt

Sind alle offenen Handles eines Nachrichtenpuffer-Objekts mit einem Aufruf von OS_SYSMSG_LCLOSE() freigegeben worden, kann der Inhalt auf eine Nachrichtendatei geschrieben werden. Die Funktion schreibt den Inhalt des Nachrichtenpuffers in eine Datei. Beim Beenden der Anwendung schließt das Betriebssystem alle geöffneten Handles für Nachrichtenpuffer-Objekte.

2.6.2.2 OS_SYSMSG_LCREATE

Die OS_SYSMSG_LCREATE()-Funktion erzeugt ein neues Nachrichtenpuffer-Objekt und liefert ein Handle, welches verwendet werden kann, um auf das Objekt zuzugreifen.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LCREATE
VAR_INPUT
    nID          : DINT;
    buffer       : ^CHAR;
    bufferSize   : UDINT;
    file_quota   : DINT;
    flags        : UDINT;
END_VAR
VAR_OUTPUT
    hLog         : UDINT;
END_VAR;

#define OS_SYSMSG_LCREATE(p1,p2,p3,p4,p5)
OS_pLslSysMsg^.lcreate
$ P_LSLSYSMSG_LCREATE(p1,p2,p3,p4,p5)
```

Übergabeparameter	Typ	Beschreibung
nID	DINT	Gibt die Identifikationsnummer des Nachrichtenpuffer-Objekts an. Der Wert muss zwischen 0 und 9 liegen. Diese Nummer wird verwendet, um den Dateinamen der Nachrichtendatei einzurichten. Wenn mehr als ein Nachrichtenpuffer verwendet wird, muss jeder Puffer eine eigene Identifikationsnummer besitzen. Folgende IDs sind für SIGMATEK Softwarekomponenten reserviert: 9,8,7 (Loader), 6 (Softwarepaket „Industrie 4.0“).
Puffer	^CHAR	Zeiger auf einen Puffer, in den die Nachrichten geschrieben werden
bufferSize	UDINT	Größe des Puffers
file_quota	DINT	Gibt die Dateiquote der Nachrichtendatei an. Ein Wert von 0 deaktiviert das Schreiben des Nachrichtenpuffers in eine Nachrichtendatei. Ein Wert von -1 deutet darauf hin, dass der System-Default Wert verwendet werden soll.

flags	UDINT	Dieser Parameter ist für zukünftige Verwendungen reserviert und muss auf 0 gesetzt werden.
Rückgabeparameter	Typ	Beschreibung
hLog	UDINT	Ist die Funktion erfolgreich, ist der Rückgabewert ein Handle auf den Nachrichtenpuffer. Andernfalls ist der Rückgabewert 0.

Eine Dateiquote von 0 ist nur sinnvoll für Betriebssystemnachrichten, wenn ein statisches RAM für den Nachrichtenpuffer verwendet wird oder ein Debug-Tool vorhanden ist, um die Nachrichten in den Puffer zu laden.

2.6.2.3 OS_SYSMSG_LFLUSH

Löscht eine Log-Puffer zu einer Log-Datei auf der Festplatte. Die Daten werden am Ende der Log-Datei eingefügt. Wenn "logging to disk" nicht aktiviert ist, liefert die Funktion einen Fehler zurück.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LFLUSH
VAR_INPUT
    hLog          : UDINT;
END_VAR
VAR_OUTPUT
    result        : ;
END_VAR;

#define OS_SYSMSG_LFLUSH(p1)
    OS_pLs1SysMsg^.lflush
    $ P_LSLSYSMSG_LFLUSH(p1)
```

Übergabeparameter	Typ	Beschreibung				
hLog	UDINT	Handle auf das Nachrichtenpuffer-Objekt				
Rückgabeparameter	Typ	Beschreibung				
result	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: right;">0</td> <td>Puffer wurde erfolgreich gelöscht oder der angegebene Puffer enthält keine Daten</td> </tr> <tr> <td style="text-align: right;">-1</td> <td>Fehler</td> </tr> </table>	0	Puffer wurde erfolgreich gelöscht oder der angegebene Puffer enthält keine Daten	-1	Fehler
0	Puffer wurde erfolgreich gelöscht oder der angegebene Puffer enthält keine Daten					
-1	Fehler					

2.6.2.4 OS SYSMSG LINFO

```

Log-Puffer-Informationen erhalten.

FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LINFO
VAR_INPUT
    hLog           : UDINT;
    buf_addr       : ^UDINT;
    pos            : ^UDINT;
    bufsize        : ^UDINT;
    n_unflushed    : ^UDINT;
END_VAR
VAR_OUTPUT
    result         : UDINT;
END_VAR;

#define OS_SYSMSG_LINFO(p1,p2,p3,p4,p5)
    OS_pLslSysMsg^.linfo
    $ P_LSLSYSMSG_LINFO(p1,p2,p3,p4,p5)

```

Übergabeparameter		Typ	Beschreibung
hLog		UDINT	Handle auf das Nachrichtenpuffer-Objekt.
buf_addr		^UDINT	Zeiger auf eine Variable, wo die Adresse des Nachrichtenpuffers gespeichert wird (optional)
pos		^UDINT	Zeiger auf eine Variable, wo die aktuelle Position im Nachrichten-Puffer gespeichert ist (optional)
bufsize		^UDINT	Zeiger auf eine Variable, in der die Größe des Nachrichtenpuffers ohne vorhergehende Steuerstruktur gespeichert wird (optional)
n_unflushed		^UDINT	Zeiger auf eine Variable, in dem die Anzahl an Byte, die nicht in den Nachrichtenpuffer geschrieben wurden, gespeichert wird (optional)
Rückgabeparameter		Typ	Beschreibung
result		UDINT	

2.6.2.5 OS_SYSMSG_OPEN

Die OS_SYSMSG_OPEN()-Funktion öffnet ein bestehendes Nachrichtenpuffer-Objekt und liefert ein Handle, das verwendet werden kann, um auf das Objekt zuzugreifen.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LOPEN
VAR_INPUT
    nID          : DINT;
END_VAR
VAR_OUTPUT
    hLog         : UDINT;
END_VAR;
```

```
#define OS_SYSMSG_LOPEN(p1)
  OS_pLslSysMsg^.lopen
  $ P_LSLSYSMSG_LOPEN(p1)
```

Übergabeparameter	Typ	Beschreibung
nID	DINT	Gibt die Identifikationsnummer des Nachrichtenpuffer-Objekts an. Der Wert muss zwischen 0 und 9 liegen.
Rückgabeparameter	Typ	Beschreibung
hLog	UDINT	Ist die Funktion erfolgreich, ist der Rückgabewert ein Handle auf den Nachrichtenpuffer. Andernfalls ist der Rückgabewert 0.

2.6.2.6 OS_SYSMSG_LPRINTFLN1-4

Druck-formatierte Daten in einen Log-Puffer. Ein neues Zeilenvorschubzeichen wird am Ende des Textes eingefügt.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LPRINTFLN
VAR_INPUT
  hLog          : UDINT;
  fAddTimestamp : UDINT;
  msg           : ^CHAR;
  lpar1         : UDINT;
  lpar2         : UDINT;
  lpar3         : UDINT;
  lpar4         : UDINT;
END_VAR
VAR_OUTPUT
  result        : UDINT;
END_VAR;

#define OS_SYSMSG_LPRINTFLN1(p1,p2,p3,p4)
  OS_pLslSysMsg^.lprintfln
  $ P_LSLSYSMSG_LPRINTFLN(p1,p2,p3,p4,0 ,0 ,0 )
#define OS_SYSMSG_LPRINTFLN2(p1,p2,p3,p4,p5)
  OS_pLslSysMsg^.lprintfln
  $ P_LSLSYSMSG_LPRINTFLN(p1,p2,p3,p4,p5,0 )
#define OS_SYSMSG_LPRINTFLN3(p1,p2,p3,p4,p5,p6)
  OS_pLslSysMsg^.lprintfln
  $ P_LSLSYSMSG_LPRINTFLN(p1,p2,p3,p4,p5,p6,0 )
#define OS_SYSMSG_LPRINTFLN4(p1,p2,p3,p4,p5,p6,p7)
  OS_pLslSysMsg^.lprintfln
  $ P_LSLSYSMSG_LPRINTFLN(p1,p2,p3,p4,p5,p6,p7)
```

Übergabeparameter	Typ	Beschreibung
hLog	UDINT	Handle auf das Nachrichtenpuffer-Objekt

fAddTimestamp	UDINT	Ein Flag, das angibt, ob ein Zeitstempel in der Nachricht eingefügt werden sollte. Ein Wert von 0 bedeutet, dass kein Zeitstempel eingefügt wird, alle anderen Werte fügen einen Zeitstempel ein.
msg	^CHAR	Ein 0-terminierter Format Steuer-String
lpar1-4	UDINT	Optionale Argumente
Rückgabeparameter	Typ	Beschreibung
result	UDINT	Der Rückgabewert ist die Anzahl der geschriebenen Zeichen, wenn ein Ausgabefehler auftritt, ohne das abschließende Null-Zeichen oder eines negativen Wertes. Wenn die Anzahl der zu schreibenden Zeichen 256 überschreitet, dann werden 256 Zeichen geschrieben. In diesem Fall sollte die Nachricht in mehrere Teile aufgeteilt werden.

OS_SYSMSG_LPRINTFLNx formatiert und druckt eine Reihe von Zeichen und Werten an den Nachrichtenpuffer. Jede Funktion wird den MSG-Formatspezifikationen entsprechend umgewandelt und ausgegeben. Das MSG-Argument hat die gleiche Syntax und Verwendung wie in der Druckfunktion der C-Programmiersprache.

2.6.2.7 OS_SYSMSG_LWRITE

Die OS_SYSMSG_LWRITE()-Funktion schreibt Daten auf einen Nachrichtenpuffer.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LWRITE
VAR_INPUT
    hLog          : UDINT;
    fAddTimestamp : UDINT;
    data          : ^CHAR;
    size          : UDINT;
END_VAR
VAR_OUTPUT
    result        : UDINT;
END_VAR;

#define OS_SYSMSG_LWRITE(p1,p2,p3,p4)
    OS_pLslSysMsg^.lwrite
    $ P_LSLSYSMSG_LWRITE(p1,p2,p3,p4)
```

Übergabeparameter	Typ	Beschreibung
hLog	UDINT	Handle auf das Nachrichtenpuffer-Objekt
fAddTimestamp	UDINT	Ein Flag, das angibt, ob ein Zeitstempel zu der Nachricht hinzugefügt werden sollte oder nicht. Ein Wert von 0 bedeutet, dass kein Zeitstempel eingefügt wird, alle anderen Werte fügen einen Zeitstempel ein.

data	^CHAR	Zeiger auf die Daten, die in den Nachrichtenpuffer geschrieben werden sollen
size	UDINT	Größe der Daten
Rückgabeparameter	Typ	Beschreibung
result	UDINT	Anzahl an Byte, die in den Nachrichtenpuffer geschrieben werden

2.6.2.8 OS_SYSMSG_WRITE_I

Schreibt einen String und bis zu vier optionale Parameter vom Typ UDINT in einen Log-Puffer.

```
FUNCTION __cdecl virtual global LWrite_I
VAR_INPUT
    hLog      : UDINT;
    pTxt      : ^CHAR;
    udParam1  : UDINT;
    udParam2  : UDINT;
    udParam3  : UDINT;
    udParam4  : UDINT;
END_VAR
VAR_OUTPUT
    result    : UDINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
hLog	UDINT	Handle auf das Nachrichtenpuffer-Objekt
pTxt	^CHAR	Ein 0-terminierter Format Steuer-String. Dieser Format-String muss Spezifikationen enthalten, die das Ausgabeformat der Argumente bestimmen. Eine Beschreibung des Format-Strings ist in der Dokumentation der printf-Funktion in C zu finden.
udParam1-4	UDINT	Optionale Argumente
Rückgabeparameter	Typ	Beschreibung
result	UDINT	Länge des Strings im Log-Puffer zurück

Ein Zeilenvorschubzeichen wird am Ende des Textes eingefügt; diese Funktion kann von einem Interrupt aufgerufen werden. Ist die Länge einer Zeichenkette größer als 31, wird diese auf eine Länge von 31 Zeichen gekürzt. Wenn ein optionaler Parameter verwendet wird, muss der String den geeigneten Druckformat-Spezifikationen entsprechen. Diese Funktionsparameter werden zunächst in einer Warteschlange von einem Interrupt Log-Puffer gespeichert und später auf die tatsächlichen Log-Puffer gedruckt. Aus diesem Grund ist zu beachten, dass ein optionaler Parameter sich nicht auf Daten referenziert, die

nur in der Interrupt-Funktion gültig sind. Ein lokales Array zum Beispiel kann nicht in dem Stack deklariert und einen Format-String "% s" zugewiesen werden.

Die Einträge aus dem Interrupt Log-Puffer werden abgearbeitet, sobald einer der sysmsg API-Funktionen aufgerufen wird. Wird diese Funktion zu oft aufgerufen, kann die interne Warteschlange überfüllt werden. In einem solchen Fall wird die älteste Nachricht verworfen. Jeder Eintrag erhält eine Nummer, die auch in das Log geschrieben wird. Dies ermöglicht den Benutzern zu sehen, ob Nachrichten verloren gegangen sind.

2.6.2.9 OS_SYSMSG_LWRITELN

Schreibt einen 0-terminierten String mit einem Zeilenwechselzeichen am Ende der Daten in einen Log-Puffer.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LWRITELN
VAR_INPUT
    hLog          : UDINT;
    fAddTimestamp : UDINT;
    txt          : ^CHAR;
END_VAR
VAR_OUTPUT
    result        : UDINT;
END_VAR;

#define OS_SYSMSG_LWRITELN(p1,p2,p3)
    OS_pLs1SysMsg^.lwrriteln
    $ P_LSLSYSMSG_LWRITELN(p1,p2,p3)
```

Übergabeparameter	Typ	Beschreibung
hLog	UDINT	Handle auf das Nachrichtenpuffer-Objekt
fAddTimestamp	UDINT	Ein Flag, das angibt, ob ein Zeitstempel in der Nachricht eingefügt werden sollte. Ein Wert von 0 bedeutet, dass kein Zeitstempel eingefügt wird, alle anderen Werte fügen einen Zeitstempel ein.
txt	^CHAR	Zeiger auf einen 0-terminierten String
Rückgabeparameter	Typ	Beschreibung
result	UDINT	Anzahl an Byte, die in den Nachrichtenpuffer geschrieben werden

2.6.3 Sysmsg – Änderungen im neuen Betriebssystem

Es gibt nur wenig Änderungen in den LasalOS-Versionen ≥ 5.80 (C-IPC 5.60 KM):

- Neuer Zeitstempel
- Log-Puffer wird automatisch gelöscht

- Neues Dateiformat.
- Erweiterter Userlog

Timestamp (Zeitstempel)

Der zur Nachricht hinzugefügte Zeitstempel enthält Millisekunden.

Logpuffer wird automatisch gelöscht

Wird der Platz im Log-Puffer sehr gering, dann wird der Puffer geleert. Diese Funktion ist nur für den System-Log Puffer (ID 0) und den erweiterten Userlog-Puffer (siehe unten) (ID 1, ID 2) verfügbar.

Dateiformat

Beim Start der Log-Datei gibt es einen SIGMATEK-spezifischen Header, der die Log-Puffer ID und die OS-Versionen enthält. Die Kopfzeile der Userlog-Dateien hat einen zusätzlichen Eintrag (siehe unten). Die Einträge jeder Zeile sind durch ein Semikolon (;) getrennt.

Erweiterter Userlog

Es gibt zwei Log-Puffer (ID 1 - niedrige Priorität und ID 2 - hohe Priorität), die vom OS reserviert werden. Daher ist das Erstellen, Öffnen oder Schließen eines Log-Puffers nicht notwendig. Die Log-Dateien werden immer als event01.log für die niedrige Priorität und als event02.log für die hohe Priorität im Log-Puffer gespeichert. Bei einem OS-Update von < 5.80 (C-IPC 5.60 KM) auf ≥ 5.80 wird die alte Log-Datei von 'eventxx.log' auf 'eventxx.err' umbenannt, weil das Betriebssystem keine gültige Header-Datei findet. Ist keine Logdatei vorhanden, erstellt das OS beim Start-up eine neue. Die Userlog-Dateien haben eine erweiterte Benutzer-Header-Datei neben der System-Header-Datei für User-Einträge. Das Betriebssystem füllt den Benutzer-Header mit einer Standardnachricht. In jeder Zeile der Header-Datei ist ein Zeitstempel vorhanden. Beim Beenden der Anwendung werden die Log-Dateien gelöscht.

2.6.4 Erweiterte Userlog Funktionen

Header files: lsl_st_sysmsg.h

2.6.4.1 OS_USERLOG_EXPORT

Kopiert alle Userlog-Dateien und die System-Eventlog-Datei (ID 0) mit Backup-Dateien in einen bestimmten Pfad.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LUSER_EXPORT
VAR_INPUT
    path      :^CHAR;
END_VAR
VAR_OUTPUT
    rc       : DINT;
END_VAR;

#define OS_USERLOG_EXPORT(p1)
    OS_pLs1SysMsg^.pLprintfln_userlog
    $ P_LSLSYSMSG_LUSER_EXPORT(p1)
```

Übergabeparameter		Typ	Beschreibung
path		^CHAR	Zielpfad
Rückgabeparameter		Typ	Beschreibung
rc		DINT	0 Funktion erfolgreich <0 Fehler-Code

Der Pfad muss ein gültiger Dateinamen-Syntax sein ("Laufwerk : \ Verzeichnis", "\ \ ...", "Directory"). Wenn path = NIL, dann werden die Dateien in das aktuelle Verzeichnis kopiert. Sind die Dateien im Zielpfad bereits vorhanden, werden diese überschrieben. Die Funktion blockiert nicht. Bis alle Dateien erfolgreich kopiert wurden oder ein Fehler auftritt, wird LUSER_BUSY zurückgeliefert. Der Rückgabewert gibt nur Informationen über den erfolgreichen Aufruf der Funktion und zum LUSERBUSY Zustand. Es wird keine Information über das erfolgreiche Kopieren ausgegeben. [OS_USERLOG_LAST_EXPORT_RESULT\(\)](#) gibt das Ergebnis des Kopiervorgangs zurück.

2.6.4.2 OS_USERLOG_FILEHEADER

Schreibt eine User-Header-Datei auf das Logfile.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LUSER_FILEHEADER
VAR_INPUT
    str      :^CHAR;
    LogFileID  : DINT;
    bNonBlocking : USINT;
END_VAR
VAR_OUTPUT
    rc       : DINT;
```

```

END_VAR;

#define OS_USERLOG_FILEHEADER(p1,p2,p3)
  OS_pLslSysMsg^.plprintfln_userlog
  $ P_LSLSYSMSG_LUSER_FILEHEADER(p1,p2,p3)

```

Übergabeparameter	Typ	Beschreibung	
str	^CHAR	Datei-Header-Nachricht als 0-terminierter String	
LogFileID	DINT	Logdatei, in die der Header geschrieben wird	
bNonBlocking	USINT	0	Funktion blockiert
		1	Funktion blockiert nicht
Rückgabeparameter	Typ	Beschreibung	
rc	DINT	0 Funktion erfolgreich <0 Fehler-Code	

Die längste Datei-Header Nachricht, die geschrieben werden kann, ist 106 Bytes groß. Wenn die Anzahl von 106 Zeichen überschritten wird, dann werden nur 106 Zeichen geschrieben. Die User-Datei Header ist nur für Log-Datei ID 1 und ID 2 verfügbar. Ist der bNonBlocking Parameter gleich 0, dann blockiert die Funktion solange sie die Datei-Header schreiben muss. Ist der bNonBlocking Parameter 1, so blockiert die Funktion nicht und liefert LUSER_BUSY zurück, bis sie den Datei-Header fertig geschrieben hat oder ein Fehler aufgetreten ist.

2.6.4.3 OS_USERLOG_FLUSH

Die Funktion überträgt den angegebenen Logpuffer auf die Festplatte.

```

FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LUSER_FLUSH
VAR_INPUT
  LogFileID      : DINT;
  bNonBlocking   : USINT;
END_VAR
VAR_OUTPUT
  rc            : DINT;
END_VAR;

#define OS_USERLOG_FLUSH(p1,p2)
  OS_pLslSysMsg^.plprintfln_userlog
  $ P_LSLSYSMSG_LUSER_FLUSH(p1,p2)

```

Übergabeparameter	Typ	Beschreibung	
-------------------	-----	--------------	--

LogFileD	DINT	LUSER_LOW_PRIO_ID oder 1 für den niederpriorenen Log-Puffer LUSER_HIGH_PRIO_ID oder 2 für den hochpriorenen Log-Puffer
bNonBlocking	USINT	0 Funktion blockiert 1 Funktion blockiert nicht.
Rückgabeparameter	Typ	Beschreibung
rc	DINT	0 Funktion erfolgreich <0 Fehler-Code

Die Funktion überträgt den Log-Puffer auf den Pfad, der mit der Umgebungsvariable APPMSGPATH eingestellt wurde oder auf den Standard-System-Pfad C:\SYSMSG. Wenn der bNonBlocking Parameter 0 ist, dann blockiert die Funktion so lange, bis die Funktion den Log-Puffer übertragen hat. Wenn der bNonBlocking Parameter 1 ist, dann blockiert die Funktion nicht und liefert LUSER_BUSY zurück, bis der Übertragungsprozess fertig ist.

2.6.4.4 OS_USERLOG_LAST_EXPORT_RESULT

Diese Funktion wird verwendet, um Informationen über den letzten OS_USERLOG_EXPORT Aufruf abzufragen.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LUSER_LAST_EXPORT_RESULT
VAR_INPUT
  PtrExpResult :^PLUSER_EXPORT_RESULT;
END_VAR
VAR_OUTPUT
  errcount : DINT;
END_VAR;
#define OS_USERLOG_LAST_EXPORT_RESULT(p1)
  OS_pLslSysMsg^.plprintfln_userlog
  $ P_LSLSYSMSG_LUSER_LAST_EXPORT_RESULT(p1)
```

Übergabeparameter	Typ	Beschreibung
PtrExpResult	^PLUSER_EXPORT_RESULT	Zeiger auf einen Pointer der auf ein vom Betriebssystem zugewiesenes Array zeigt, welches das Exportergebnis speichert
Rückgabeparameter	Typ	Beschreibung
errcount	DINT	≥0 Anzahl von Fehlern, die beim Kopiervorgang aufgetreten sind <0 Negativer Fehler-Code

Beispiel

```

ExportResult :^LUSER_EXPORT_RESULT;
OS_USERLOG_LAST_EXPORT_RESULT(#ExportResult) ;

TYPE
LUSER_EXPORT_RESULT: STRUCT
rc : DINT; // Fehler-Code aus den Dateifunktionen
LogFileID : DINT; // ID aus der Logdatei
IsBackupFile : USINT; // 0 .. logfile (.log), 1 .. backupfile (.bak)
pNext :^LUSER_EXPORT_RESULT;
END_STRUCT;
PLUSER_EXPORT_RESULT :^LUSER_EXPORT_RESULT;
END_TYPE

```

Das Array wird mit allen Ergebnissen gefüllt, auch wenn kein Fehler aufgetreten ist. Verwenden Sie den Zeiger pNext, um den nächsten Eintrag abzurufen, bis pNext = NIL. Isl_st_osfile den DateiFehler-Code beschreibt.

rc: -1000 .. dest = NIL interner Fehler.

-1001 .. Zu wenig Arbeitsspeicher zum Allozieren eines internen Puffers, um die Dateien & zu kopieren.

2.6.4.5 OS_USERLOG_PRINTFNLN (1-2)

Schreibt eine Nachricht an den niederpriorenen Log-Puffer.

```

FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LUSER_PRINTFNLN
VAR_INPUT
msg :^CHAR;
param1 : DINT;
param2 : DINT;
END_VAR
VAR_OUTPUT
rc : DINT;
END_VAR;

#define OS_USERLOG_PRINTFNLN(p1)
OS_pLslSysMsg^.plprintfln_userlog
$ P_LSLSYSMSG_LUSER_PRINTFNLN(p1, 0, 0)
#define OS_USERLOG_PRINTFNLN1(p1,p2)
OS_pLslSysMsg^.plprintfln_userlog
$ P_LSLSYSMSG_LUSER_PRINTFNLN(p1,p2, 0)
#define OS_USERLOG_PRINTFNLN2(p1,p2,p3)
OS_pLslSysMsg^.plprintfln_userlog
$ P_LSLSYSMSG_LUSER_PRINTFNLN(p1,p2,p3)

```

Übergabeparameter	Typ	Beschreibung
-------------------	-----	--------------

msg	^CHAR	0-terminierter Format Steuer-String
param1-2	DINT	Optionale Argumente
Rückgabeparameter	Typ	Beschreibung
rc	DINT	>0 Anzahl geschriebenen Byte <0 Fehler-Code (siehe Isl_st_sysmsg.h)

Die längste Nachricht, die geschrieben werden kann beträgt 234 Bytes. Wenn die Zahl der Zeichen 234 überschreitet, werden nur 234 Zeichen geschrieben. Wird der Platz im Log-Puffer zu gering, dann löscht das Betriebssystem den Log-Puffer. Die Funktion blockiert nicht, liefert aber LUSER_BUSY während des Vorganges. In diesem Fall werden keine Nachrichten eingeloggt, bis der Ladeprozess abgeschlossen ist.

2.6.4.6 OS_USERLOG_WRITEEVENT

Die Funktion schreibt 5 Werte in Log-Puffer mit hohen Priorität.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LUSER_WRITEEVENT
VAR_INPUT
  hID      : USINT;
  scancode : UINT;
  picnum   : UINT;
  X        : UINT;
  Y        : UINT;
END_VAR
VAR_OUTPUT
  rc      : DINT;
END_VAR;

#define OS_USERLOG_WRITEEVENT(p1,p2,p3,p4,p5)
  OS_pLslSysMsg^.plprintfln_userlog
  $ P_LSLSYSMSG_LUSER_WRITEEVENT(p1,p2,p3,p4,p5)
```

Übergabeparameter	Typ	Beschreibung
hID-Y		Zu schreibende Werte
Rückgabeparameter	Typ	Beschreibung
rc:	DINT	>0 Anzahl geschriebenen Byte <0 Fehler-Code (siehe Isl_st_sysmsg.h)

Wird der Platz im Log-Puffer zu gering, dann löscht das Betriebssystem den Log-Puffer. Die Funktion blockiert nicht, liefert aber LUSER_BUSY beim Überladen zurück. In diesem

Fall werden keine Nachrichten eingeloggt, bis der Ladeprozess abgeschlossen ist. Diese Funktion kann aus dem REALTIME Task aufgerufen werden.

2.7 API TCP Benutzer-Schnittstelle

2.7.1 Fehlerwerte

Kann keinen Fehler-String zurückliefern

TCP_NOT_READY	-2000	<p>Nicht bereit.</p> <p>Beispiele:</p> <p>OS_TCP_USER_CONNECT(), OS_TCP_USER_ACCEPT(): es wurde noch keine Verbindung hergestellt.</p> <p>OS_TCP_USER_RECV(): es sind keine Empfangsdaten vorhanden.</p> <p>OS_TCP_USER_SEND(): der Remote Host ist derzeit nicht bereit Daten zu empfangen.</p>
TCP_INVALID_SOCKET_STATE	-2001	<p>Socket ist im falschen Zustand.</p> <p>Beispiele:</p> <p>Beim Aufruf von OS_TCP_USER_CONNECT() befindet sich der Socket nicht im richtigen Zustand.</p> <p>OS_TCP_USER_NREAD() oder OS_TCP_USER_IOCTLSOCKET() oder OS_TCP_USER_RECV() oder OS_TCP_USER_SEND() etc. werden aufgerufen, ohne vorher OS_TCP_USER_ACCEPT() oder OS_TCP_USER_CONNECT() aufgerufen zu haben.</p> <p>OS_TCP_USER_ACCEPT() ohne vorheriges OS_TCP_USER_LISTEN().</p>
TCP_NOMEM_ERROR	-2002	<p>Zu wenig freier Speicher.</p> <p>Beispiele:</p> <p>OS_TCP_USER_SOCKET(): es wurden zu viele Sockets erzeugt.</p> <p>OS_TCP_USER_STRTOULONG_ASY: interner Fehler.</p>
TCP_BUFFER_TO_SMALL	-2003	<p>Der Zielpuffer ist zu klein.</p> <p>Beispiele:</p> <p>OS_TCP_USER_TOIP(), OS_TCP_USER ULONGTOSTR(), OS_TCP_USER_GETERRORSTRING().</p> <p>OS_TCP_USER_STRTOULONG_ASY: die Mailbox ist voll (interner Fehler).</p>
TCP_MAXSOCKETS_ERROR	-2004	Alle Sockets sind belegt.

TCP_INVALID_SOCK_NUM	-2005	Die übergebene Socket-Nummer ist ungültig.
TCP_INVALID_NUMBER	-2007	<p>Ungültiger Eingabewert. Beispiel: Eine Komponente einer binären IP-Adresse ist größer als 255: OS_TCP_USER_TOIP().</p>
TCP_SYSTEM_ERROR	-2008	Fehler beim Lesen der Socket-Info.
TCP_NOIF_ERROR	-2009	Nummer der Netzwerkschnittstelle ungültig. Z.B. OS_TCP_USER_IPINFO().

Kann einen Fehler-String zurückliefern

TCP_ADDR_NOT_AVAILABLE	-3000	endpoint address not available
TCP_ADDR_IN_USE	-3001	a socket is already bound to this address
TCP_AF_NO_SUPPORT	-3002	family not supported
TCP_ARP_TABLE_FULL	-3003	ARP table full
TCP_INVALID_BAUD	-3004	invalid baud rate
TCP_INVALID_COMM_PORT	-3005	invalid comm port number
TCP_INVALID_DEVICE	-3006	invalid device type
TCP_INVALID_IFACE	-3007	invalid interface number
TCP_INVALID_MASK	-3008	invalid mask (ether must not be all fs)
TCP_INVALID_PING	-3009	invalid ping response
TCP_CONN_REFUSED	-3010	endpoint refused connection
TCP_DEST_ADDR_REQ	-3011	destination address is required
TCP_DEST_UNREACHABLE	-3012	destination is unreachable (ICMP)

TCP_INVALID_PARA M	-3013	invalid parameter (pointer is 0, etc.)
TCP_IFACE_CLOSE D	-3014	interface closed
TCP_IFACE_TABLE_ FULL	-3015	interface table full
TCP_IFACE_OPEN_ FAIL	-3016	interface open failed
TCP_IN_PROGRESS	-3017	operation (connect) is in progress
TCP_INVALID_FUNC	-3018	invalid function call (parameter)
TCP_SOCKET_CON NECTED	-3019	socket is already connected
TCP_MC_TABLE_FU LL	-3020	multicast table full
TCP_MC_ADDR_NO T_FOUND	-3021	multicast address not found
TCP_OUT_OF_POR TS	-3022	out of ports
TCP_NET_DOWN	-3023	network is down (send failed)
TCP_NET_UNREAC H	-3024	network unreachable (keepalive failed)
TCP_OUT_OF_DCU S	-3025	out of DCUs (packets)
TCP_OPTPARAM_IN VALID	-3026	option parameter is invalid
TCP_SOCK_NOT_C ONNECTED	-3027	socket is not connected
TCP_RTIP_NOT_INS T	-3028	RTIP not initialized (i.e. xn_rtip_init not called)
TCP_INVALID_SOCK ET	-3029	invalid socket descriptor
TCP_NUM_DEVICE	-3030	not enough devices
TCP_OP_NOT_SUP PORT	-3031	socket type or specified operation not supported for this function
TCP_OUTPUT_FULL	-3032	send failed due to output list being full

TCP_PROBE_FAIL	-3033	could not determine device
TCP_RENTRANT	-3034	a non-reentrant function was reentered
TCP_ROUTE_NOT_FOUND	-3035	routing table entry not found
TCP_ROUTE_FULL	-3036	routing table full
TCP_RSC_INIT_FAIL	-3037	resource initialization failed (signals, semaphores, tasks)
TCP_SHUTDOWN	-3038	illegal operation due to socket shutdown
TCP_TIMEOUT	-3039	timeout
TCP_TYPE_NOT_SPPORT	-3040	type not supported (only SOCK_STREAM and SOCK_DGRAM are supported)
TCP_WOULD_ARP	-3041	send needs to ARP but ARP is disabled
TCP_WOULD_BLOCK	-3042	socket non-blocking but function would block
TCP_UNKNOWN_ERROR	-3043	unknown error

Wird das TCP User Interface unter Lars verwendet, so können zusätzlich Windows Socket Error Codes (WSA) zurück geliefert werden. Hierbei ist zu beachten, dass diese negativ und mit einem Offset von 4000 versehen sind. Bsp.: Lars -14040 => WSA 10040 (WSAEMSGSIZE)

Details: <http://msdn.microsoft.com/en-us/library/windows/desktop/ms740668%28v=vs.85%29.aspx>

2.7.2 Anwendung der OS-Funktionen

Zur Verwendung der OS TCP USER FUNCTIONS muss ein Pointer namens `Isl_tcp_user` vom Typ `LSL_TCP_USER` definiert und in `Isl_st_tcp_user.h` angegeben werden.

Der Pointername muss genau wie oben geschrieben werden, da die in der Datei `Isl_st_tcp_user.h` angegebenen Makros diesen Zeiger verwenden, um die OS-Funktionen aufzurufen.

Bevor die OS TCP USER FUNCTIONS benutzt werden können, muss der Zeiger initialisiert werden.

Dieser verwendet das "OS_CILGet" OS-Makro, welches in der globalen Header-Datei `rtos_interfaces.h` angegeben ist.

Der Name des OS TCP Benutzeroberfläche ist "TCP_USER". Dieser Name wird für den ersten Parameter des OS_CILGet Makros benötigt.

Ist die Schnittstelle in der verwendeten OS-Version vorhanden wird der Pointer mit den Adressen der Betriebssystem-Funktionen initialisiert. Ansonsten wird ein NIL-Pointer vom zweiten Parameter des OS_CILGet Makros zurückgegeben. Der Pointer lsl_tcp_user vom Typ LSL_TCP_USER kann als eine globale Variable oder eine Mitgliedsvariable definiert werden.

Um eine Member-Variable vom Typ LSL_TCP_USER zu erstellen, muss die "lsl_st_tcp_user.h" Header-Datei in das LASAL CLASS-Projekt eingefügt und die globale Flag in die Header-Datei gesetzt werden.

Beispiel

```
FUNCTION VIRTUAL GLOBAL MyClass::Init

IF (_firstscan) THEN
    OS_CILGet("TCP_USER", #lsl_tcp_user);
END_IF;

END_FUNCTION //VIRTUAL GLOBAL GetIP::Init
```

Sobald das OS TCP USER INTERFACE in der verwendeten Betriebssystem-Version vorhanden ist, kann das OS_TCP_USER_VERSION Makro verwendet werden, um zu überprüfen, ob eine bestimmte Funktion zur Verfügung steht.

Siehe die Anforderungen für jedes Makro.

Beispiel

```
IF (lsl_tcp_user <> NIL) THEN
    IF (OS_TCP_USER_VERSION >= 4) THEN
        // function available
        OS_TCP_USER_IPINFO(...);
    ELSE
        // function not available
    END_IF;
ELSE
    // interface not available
END_IF;
```

2.7.3 OS TCP USER-FUNKTIONEN

2.7.3.1 OS_TCP_USER_ACCEPT

Eine Verbindung von einem beliebigen Remote-Host akzeptieren.

```
VAR_INPUT
    socket      : DINT;
    timeout_ms : UDINT;
END_VAR
```

```
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
socket		DINT	Socket-Nummer
timeout_ms	UDINT	0	Funktion blockiert nicht
		>0	Timeout in ms, Funktion blockiert
Rückgabeparameter		Typ	Beschreibung
retval		DINT	≥ 0 Anschlussnummer, die dieser Verbindung zugeordnet ist <0 Fehler-Code

Wenn timeout_ms auf >0 gesetzt ist, wird die Funktion blockiert, bis die Verbindung hergestellt, ein Fehler aufgetreten ist oder wenn die Funktion zu lange dauert und das Timeout abgelaufen ist.

Ist timeout_ms auf 0 gesetzt, liefert die Funktion TCP_NOT_READY zurück, bis die Verbindung hergestellt oder ein Fehler aufgetreten ist (die Funktion muss abgefragt werden).

Beispiel

```
retval      : DINT;
newsocket   : DINT;

retval := OS_TCP_USER_ACCEPT(socket, 0);
if retval < 0 & retval >> TCP_NOT_READY then
    ...error
else
    newsocket := retval; ...newsocket accepted ... ready to send or receive data
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.2 OS_TCP_USER_CLOSESOCKET

Einen bestehenden Anschluss schließen.

```
VAR_INPUT
  socket      : DINT;
  type        : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
socket	DINT	Socket-Nummer	
typ	UDINT	0	Hard close
		>0	Soft close
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0 Funktion erfolgreich	
		<0 Fehler-Code	

Ein Hard-Close schließt den Anschluss sofort und alle nicht gesendeten Daten gehen verloren.

Eine Soft-Close schließt den Anschluss, ausstehende Daten werden aber gesendet.

Beispiel

```
retval: DINT;

retval := OS_TCP_USER_CLOSESOCKET(socket, 0);
if retval < 0 then
  ...error
  else
  ...o.k.
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.3 OS_TCP_USER_CONNECT

Erstellt eine Verbindung mit einem bestimmten Anschluss.

```

VAR_INPUT
  socket      : DINT;
  localport   : UDINT;
  IPAddress  : ^CHAR;
  port        : UDINT;
  timeout_ms : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;

```

Übergabeparameter		Typ	Beschreibung
socket		DINT	Socket-Nummer
localport		UDINT	Lokale Port-Nummer (falls 0, wird eine einzigartige Port-Nummer zugewiesen)
IPAddress		^CHAR	Ziel IP-Adresse
port		UDINT	Ziel Port-Nummer
timeout_ms		UDINT	0 Funktion blockiert nicht >0 Timeout in ms, Funktion blockiert
Rückgabeparameter		Typ	Beschreibung
retval		DINT	0 Funktion erfolgreich <0 Fehler-Code

Wurde der Aufruf des Anschlusses erfolgreich abgeschlossen, so ist dieser zum Senden und Empfangen von Daten bereit.

Jede Verbindung benötigt einen anderen lokalen Port.

Wenn timeout_ms > 0 ist, wird die Funktion blockiert bis die Verbindung hergestellt, ein Fehler aufgetreten ist oder wenn die Funktion zu lange dauert und das Timeout abgelaufen ist.

Ist timeout_ms auf 0 gesetzt, liefert die Funktion TCP_NOT_READY zurück, bis die Verbindung hergestellt oder ein Fehler aufgetreten ist (die Funktion muss abgefragt werden).

Beispiel

```

retval : DINT;

retval := OS_TCP_USER_CONNECT(socket, 1000, "192.168.44.105", 21, 0);
if retval < 0 & retval >> TCP_NOT_READY then
    ...error
else
    ...connected...
end_if;

```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.4 OS_TCP_USER_GETCOMLINKPORT

Gibt die Portnummer zurück, die COMLink verwenden soll, um auf eingehende Verbindungen zu warten.

```

VAR_OUTPUT
    port : UDINT;
END_VAR;

```

Rückgabeparameter	Typ	Beschreibung
retval	DINT	≥0 Port-Nummer (1955 ist der erste Comlink Port)

2.7.3.5 OS_TCP_USER_GETERRORSTRING

Damit erhält man eine TCP-Benutzer-Fehlermeldung.

```

VAR_INPUT
    buffer : ^CHAR;
    buflen : UDINT;
    errvalue : DINT;
END_VAR
VAR_OUTPUT
    retval : DINT;
END_VAR;

```

Übergabeparameter	Typ	Beschreibung

buffer	^CHAR	Zeiger auf einen Puffer, der die Fehlermeldung erhält
buflen	UDINT	Länge des Puffers (≥ 90)
errvalue	DINT	Negativer Wert, der von der Funktion zurückgegeben wird
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich <0 Fehler-Code

Beispiel

```

retval      : DINT;
errvalue    : DINT;
errstring   : ARRAY[0..90] of CHAR;

retval := OS_TCP_USER_CONNECT(socket, 1000, "192.168.44.0", 21);
if retval < 0 & retval >> TCP_NOT_READY then
  errvalue := retval;

  retval := OS_TCP_USER_GETERRORSTRING(#errstring[0], sizeof(errstring), errvalue);
  if retval < 0 then
    ...error
  else
    ...got error message
  end_if;

else
  ...connected...
end_if;

```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.6 OS_TCP_USER_GETLINKSTATUS

Die Funktion liefert Informationen über die Ethernet-Schnittstelle iface in der Form von Bits.

```

VAR_INPUT
  iface      : DINT;
  pLinkStatus : ^UDINT;
END_VAR

```

```
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
iface	DINT	Nummer der Ethernet-Schnittstelle (0 für die erste Schnittstelle, 1 für die zweite)	
pLinkStatus	^UDINT	Zeiger auf den Puffer für den Link-Status	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	≥0 Kein Fehler <0 Fehler-Code	

Bedeutung der Bits im Link-Status-Puffer:

Bit3 = 1	Übertragungsgeschwindigkeit 1 Gigabit pro Sekunde
Bit3 = 0 und Bit1 = 1	Übertragungsgeschwindigkeit 100 Megabit pro Sekunde
Bit3 = 0 und Bit1 = 0	Übertragungsgeschwindigkeit 10 Megabit pro Sekunde
Bit2 = 1	Full Duplex Mode
Bit2 = 0	Half Duplex Mode
Bit0 = 1	Verbindung hergestellt
Bit0 = 0	Keine Verbindung hergestellt

In der Header-Datei Isl_st_tcp_user.h stehen einige #defines zur Auswertung der Bits zur Verfügung.

#define IP_LINK_STATUS	0x01	
		0 Keine Verbindung
		1 Verbindung hergestellt
#define IP_WIRE_SPEED	0x02	
		0 10 Mbits/s
		1 100 Mbit/s
#define IP_DUPLEX_MODE	0x04	
		0 Half duplex

1 Full duplex

2.7.3.7 OS_TCP_USER_GETPEERIP

Damit erhält man die IP-Adresse des Remote-Host der mit dem Sockel verbunden ist.

```
VAR_INPUT
    socket      : DINT;
    sin_addr    : ^UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
socket		DINT	Socket-Nummer
sin_addr		^UDINT	Zeiger auf eine unsigned long Variable, die die IP-Adresse erhält
Rückgabeparameter		Typ	Beschreibung
retval		DINT	0 Funktion erfolgreich <0 Fehler-Code

Beispiel

```
retval  : DINT;
ipaddr  : UDINT;

retval := OS_TCP_USER_GETPEERIP(socket, #ipaddr);
if retval < 0 then
    ...error
else
    ...o.k.
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In Isl_st_tcp_user.h angegeben

2.7.3.8 OS_TCP_USER_GETPEERPORT

Man erhält die Port-Nummer des Remote-Host, der mit dem Sockel verbunden ist.

```
VAR_INPUT
    socket      : DINT;
    sin_port    : ^UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
socket	DINT	Socket-Nummer	
sin_port	^UDINT	Zeiger auf eine unsigned long Variable, die die Port-Nummer erhält	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	Funktion erfolgreich
		<0	Fehler-Code

Beispiel

```
retval  : DINT;
port    : UDINT;

retval := OS_TCP_USER_GETPEERIP(socket, #port);
if retval < 0 then
    ...error
else
    ...o.k.
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In Isl_st_tcp_user.h angegeben

2.7.3.9 OS_TCP_USER_GETSERVBYNAME

Man erhält Service-Info von Service- und Protokollname. Diese Funktion steht nur beim Betriebssystem RTK zur Verfügung.

```
VAR_INPUT
```

```

name      : ^CHAR;
proto     : ^CHAR;
END_VAR
VAR_OUTPUT
    retval  : UDINT;
END_VAR

```

Übergabeparameter		Typ	Beschreibung
name		^CHAR	Zeiger auf einen 0-terminierten Servicenamen
proto		^CHAR	Optional Zeiger auf einen 0-terminierten Protokollnamen
Rückgabeparameter		Typ	Beschreibung
retval		UDINT	0 Funktion nicht erfolgreich >0 Funktion erfolgreich, die Port-Nummer des Service wird zurückgegeben

Wenn proto = NIL, liefert OS_TCP_USER_GETSERVBYNAME den ersten Service-Eintrag, wo der Name stimmt. Ansonsten gleicht GETSERVBYNAME den Namen und das Protokoll ab.

Die Namen werden ohne Berücksichtigung der Groß- und Kleinschreibung verglichen.

Service-Name	Protokoll-Name	Beschreibung	Version
comlink_clnt	TCP	Basis Portnummer für Clients / z.B. Alarm im LSE-Kernel	Ab 01.01.004
online	TCP	Online-Server	Ab 01.01.004
comlink	TCP	Comlink command server	Ab 01.01.004
comlink_refr	TCP	Comlink refresh server	Ab 01.01.004
Alarm	TCP	Alarm im LSE-Kernel	Ab 01.01.004

Beispiel

```

portnr      : UDINT;

portnr := OS_TCP_USER_GETSERVBYNAME("online", "tcp");

```

Anforderungen

Version: Ab LasalOS 01.01.004

LSL_TCP_USER STRUCT Member ab udVersion 3

Header: In lsl_st_tcp_user.h angegeben

2.7.3.10 OS_TCP_USER_GETSOCKIP

Man erhält die IP-Adresse des lokalen Host.

```
VAR_INPUT
  socket      : DINT;
  sin_addr    : ^UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
socket	DINT	Socket-Nummer	
sin_addr	^UDINT	Zeiger auf eine unsigned long Variable, die die IP-Adresse erhält	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0 Funktion nicht erfolgreich >0 Funktion erfolgreich, der Protokollname des Service wird zurückgegeben	

Beispiel

```
retval  : DINT;
ipaddr  : UDINT;

retval := OS_TCP_USER_GETPEERIP(socket, #ipaddr);
if retval < 0 then
  ...error
else
  ...o.k.
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.11 OS_TCP_USER_GETSOCKPORT

Man erhält die Portnummer des lokalen Host.

```
VAR_INPUT
    socket      : DINT;
    sin_port    : ^UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
socket	DINT	Socket-Nummer
sin_port	^UDINT	Zeiger auf eine unsigned long Variable, die die Port-Nummer erhält
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich <0 Fehler-Code

Beispiel

```
retval  : DINT;
port    : UDINT;

retval := OS_TCP_USER_GETPEERIP(socket, #port);
if retval < 0 then
    ...error
else
    ...o.k.
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.12 OS_TCP_USER_IOCTLSOCKET

Die Aufgaben dieser Funktion werden mit dem Parameter select_type ausgewählt:

Parameter select_type ist READ_AVAILABLE (1): die Funktion liefert die Anzahl der zum Lesen verfügbaren Bytes.

Parameter select_type ist SET_SOCKET_MODE (3): die Funktion setzt den Socket in den nicht-blockierenden (onoff=1) oder blockierenden (onoff=0) Modus.

Die #defines READ_AVAILABLE und SET_SOCKET_MODE stehen in der Header-Datei Isl_st_tcp_user.h zur Verfügung.

```
VAR_INPUT
  Socket      : DINT;
  select_type : UDINT;
  onoff       : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
socket	DINT	Socket-Nummer	
select_type	UDINT	READ_AVAILABLE (1): Liefert die Anzahl der zum Lesen verfügbaren Bytes. Bei TCP ist dies die Anzahl der Bytes im Empfangsfenster. Bei UDP ist es die Anzahl der Bytes im ersten Paket am UDP-Eingang. WRITE_AVAILABLE (2): Nur bei RTK. Liefert die Anzahl der zum Schreiben verfügbaren Bytes, ohne zu blockieren. Bei TCP ist dies die Größe des Sendefensters abzüglich der Anzahl der Bytes in der Warteschlange. Bei UDP ist dies die maximale Größe einer Sendung, d. h. die maximale Fragmentgröße, wenn Fragmente aktiviert sind, oder die maximale Datenmenge, die in ein Paket passt, wenn die Fragmentierung deaktiviert ist. SET_SOCKET_MODE (3): Setzt den Socket in den nicht-blockierenden oder blockierenden Modus. Siehe Parameter onoff.	
onoff	UDINT	0 Blocking mode 1 Non-blocking mode	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	≥0 Kein Fehler <0 Fehler-Code	

Die #defines READ_AVAILABLE, WRITE_AVAILABLE und SET_SOCKET_MODE für den Parameter select_type stehen in der Header-Datei lsl_st_tcp_user.h zur Verfügung.

2.7.3.13 OS_TCP_USER_IPINFO

Rückgabeinformationen über die angegebene Netzwerkschnittstelle.

```
VAR_INPUT
  iface      : DINT;
  option     : DINT;
  pErg       : ^UDINT;
END_VAR
VAR_OUTPUT
  retval     : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
iface		DINT	Nummer der Netzwerkschnittstelle, um Informationen zu erhalten
Option		DINT	Informationen Option
pErg		^UDINT	Informationen
Rückgabeparameter		Typ	Beschreibung
retval		DINT	0 Funktion erfolgreich <0 Fehler-Code

Option	pErg	OS Version
IP_OPT_ADDR	Binäre IP-Adresse	≥ 01.01.004
IP_OPT_SUBNETMASK	Binäre Subnetzmaske	≥ 01.01.004
IP_OPT_ETHERNET_ADDR	Adresse des Speicherorts, an dem die Ethernet-Adresse gespeichert ist. Um die Ethernet-Adresse zu erhalten, muss ein USINT-Zeiger auf diese Adresse gecastet werden.	≥ 01.01.021
IP_OPT_GATEWAY	Binäre Gateway-Adresse	≥ 01.01.127
IP_OPT_PORT	TCP-Port, auf welchem der TCP-Server hört	≥ 01.02.170
IP_OPT_DNS1	IP-Adresse des ersten DNS-Servers	≥ 01.02.190
IP_OPT_DNS2	IP-Adresse des zweiten DNS-Servers	≥ 01.02.190
IP_OPT_DNS3	IP-Adresse des dritten DNS-Servers	≥ 01.02.190
IP_OPT_DNS4	IP-Adresse des vierten DNS-Servers	≥ 01.02.190

Beispiel

```
ipaddr          : UDINT;
subnetmask      : UDINT;

rc := OS_TCP_USER_IPINFO(interface, IP_OPT_ADDR, #ipaddr);
if rc < 0 then
  // ... error
else
  // ... ok
end_if;

rc := OS_TCP_USER_IPINFO(interface, IP_OPT_SUBNETMASK, #subnetmask);
if rc < 0 then
  // ... error
else
  //... ok
end_if;

// OS_TCP_USER_ULONGTOSTR can be used to get the addresses in dotted format

AddressOfMAC    : UDINT;
pToMAC          : ^CHAR;
i               : USINT;

AddressOfMAC := 0;
rc := OS_TCP_USER_IPINFO(interface, IP_OPT_ETHERNET_ADDR, #AddressOfMAC);
if rc < 0 then
  // ... error
else

  // AddressOfMAC contains the address of the MAC-Address
  // Cast a pointer to this address
  pToMAC := AddressOfMAC^CHAR;

  if pToMAC <> NIL then

    for i := 0 to 5 do

      ethernet_addr[i] := pToMAC^;
      pToMAC += 1;

    end_for;

  end_if;

end_if;
```

Anforderungen

Version: Ab LasalOS 01.01.004

LSL_TCP_USER STRUCT Member ab udVersion 4

Header: In lsl_st_tcp_user.h angegeben

2.7.3.14 OS_TCP_USER_LISTEN

Versetzt einen Socket in einen Zustand, in dem er auf eine eingehende Verbindung wartet.

```
VAR_INPUT
    socket      : DINT;
    localport   : UDINT;
    backlogsize : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
socket	DINT	Socket-Nummer	
localport	UDINT	Lokale Port-Nummer	
backlogsize	UDINT	Maximale Anzahl der erlaubten anstehenden Verbindungen	
Rückgabeparameter		Typ	Beschreibung
retval	DINT		0 Funktion erfolgreich <0 Fehler-Code

Die lokale Port-Nummer ist die Port-Nummer mit dem die Clients eine Verbindung herstellen können.

Ein Anschluss im Listenmodus kann nicht zum Senden oder Empfangen von Daten verwendet werden.

Wenn Backlog-Größe einen unzulässigen Wert enthält (weniger als 1 oder größer als die Anzahl der max. Verbindungen), wird es auf die maximale erlaubte Anzahl der Benutzerverbindungen gesetzt.

Beispiel

```
retval : DINT;

retval := OS_TCP_USER_LISTEN(socket, 8000, 10);
if retval < 0 then
    ...error
else
```

```
...o.k. listen...
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.15 OS_TCP_USER_NREAD_AVAILABLE

Liefert die Anzahl von Bytes zurück, die zum Lesen verfügbar sind.

```
VAR_INPUT
    socket      : DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
socket	DINT	Socket-Nummer	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	>0	Liefert die Anzahl von Byte zurück, die zum Lesen verfügbar sind
		0	Keine verfügbaren Byte
		<0	Fehler-Code

Beispiel

```
retval      : DINT;
BytesAvail  : DINT;

retval := OS_TCP_USER_NREAD_AVAILABLE(socket);
if retval < 0 then
    ...error
else
    BytesAvailable := retval;
    ...
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.16 OS_TCP_USER_PING

Sendet einen PING an die angegebene IP-Adresse.

```
VAR_INPUT
    ipaddr      : ^CHAR;
    bytes       : UDINT;
    ttl         : UDINT;
    wait        : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
ipaddr	^CHAR	Zeiger auf Nullterminierten ASCII-String der IP-Adresse
bytes	UDINT	Größe des PING-Packets in Byte
ttl	UDINT	Lebensdauer (ttl - Time to live) des PING Packets in sec.
wait	UDINT	Timeoutzeit (rtt - Round-trip time) bis zum Eintreffen einer Antwort in ms.
Rückgabeparameter	Typ	Beschreibung
retval	DINT	≥ 0 Zeit in ms für die Antwort der Gegenstelle <0 Ping war nicht erfolgreich (Fehler-Code)

Vorgeschlagene Beispielwerte für den Ping-Aufruf:

Die vorgeschlagenen Werte beziehen sich auf die in der Command Line verwendeten Parameter-Werte für den PING-Befehl.

bytes	32
ttl [s]	500
wait [ms]	2000

Beispiel

```

bytes  : UDINT;
ttl    : UDINT;
wait   : UDINT;
dRC   : DINT;

dRC := OS_TCP_USER_PING("10.10.57.72",bytes,ttl,wait);

if dRC < 0 then
    ...error
else
    ...o.k.
end_if;

```

2.7.3.17 OS_TCP_USER_RECV

Daten auf einem Anschluss empfangen.

```

VAR_INPUT
    socket      : DINT;
    buffer      : ^CHAR;
    buflen      : UDINT;
    flags       : UDINT;
    timeout_ms : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;

```

Übergabeparameter	Typ	Beschreibung	
socket	DINT	Socket-Nummer	
buffer	^CHAR	Puffer für eingehende Daten	
buflen	UDINT	Länge des Puffers	
flags	UDINT	Nicht verwendet, muss auf 0 gesetzt werden	
timeout_ms	UDINT	0 Funktion blockiert nicht >0 Timeout in ms, Funktion blockiert	
Return parameters	Typ	Beschreibung	
retval	DINT	>0 Anzahl der verfügbaren Byte im Puffer 0 Ende der Datei (die Verbindung wurde vom Remote geschlossen).	

		<0 Fehler-Code
--	--	--------------------------

Wenn keine Daten zur Verfügung stehen, liefert die Funktion TCP_NOT_READY zurück (wenn timeout_ms auf 0 gesetzt ist).

Wenn timeout_ms > 0 ist und keine Daten zur Verfügung stehen wird die Funktion blockiert, bis Daten verfügbar sind, ein Fehler auftritt oder das Zeitlimit abgelaufen ist.

Beispiel

```
retval : DINT;
buffer : ARRAY[0..1024] of CHAR;

retval := OS_TCP_USER_RECV(socket, #buffer[0], sizeof(buffer), 0, 0);
if retval <= 0 & retval >< TCP_NOT_READY then
    ...error
elsif retval > 0 then
    ...received...
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.18 OS_TCP_USER_RECVFROM

Empfangen von Daten auf einem Socket (TCP und UDP).

```
VAR_INPUT
    socket      : DINT;
    buffer      : ^CHAR;
    buflen      : UDINT;
    flags       : UDINT;
    timeout_ms : UDINT;
    pIPAddr    : ^UDINT;
    pPort       : ^UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
socket	DINT	Socket-Nummer

buffer	^CHAR	Puffer für eingehende Daten
buflen	UDINT	Maximale Anzahl von zu erhaltenden Bytes
flags	UDINT	Nicht verwendet, muss auf 0 gesetzt werden
timeout_ms	UDINT	0 Funktion blockiert nicht >0 Timeout in ms, Funktion blockiert
pIPAddr	^UDINT	IP-Adresse des Absenders
pPort	^UDINT	Port-Nummer des Absenders
Rückgabeparameter	Typ	Beschreibung
retval	DINT	>0 Anzahl der verfügbaren Byte im Puffer 0 Ende der Datei (die Verbindung wurde vom Remote geschlossen, nur TCP) <0 Fehler-Code

Die Funktion wirkt wie [OS_TCP_USER_RECV](#). Der einzige Unterschied sind die Absenderinformationen, die von pIPAddr und PPORt zurückgeschickt werden.

Beispiel Empfangen eines UDP-Datagramms

```

socket  : DINT;
rc      : DINT;
buffer  : ARRAY [0..1023] OF CHAR;
ipaddr  : UDINT;
port    : UDINT;
mode    : UDINT;

socket := OS_UDP_USER_SOCKET();
if (socket >= 0) then

  if (mode = 1) then
    // receive all packets
    rc := OS_UDP_USER_BIND(socket, IP_ADDR_ANY, 4321);
  elsif (mode = 2) then
    // receive only broadcast packets
    rc := OS_UDP_USER_BIND(socket, IP_ADDR_BROADCAST, 4321);
  else
    // receive only packets for this interface, this interface has
    // the IP address 200.100.100.100
    ipaddr := OS_TCP_USER_STRTOULONG("200.100.100.100");

    // only packets with destination address 200.100.100.100 will be
    // received
  
```

```

rc := OS_UDP_USER_BIND(socket, ipaddr, 4321);
end_if;

if (rc = 0) then

  rc := OS_TCP_USER_RECVFROM(socket,
                               #buffer[0],
                               sizeof(buffer),
                               0,
                               2000,
                               #ipaddr,
                               #port);
  if (rc < 0)
    // error ...
  end_if;

end_if;

OS_TCP_USER_CLOSESOCKET(socket, 0);

end_if;

```

Anforderungen

Version: Ab LasalOS 01.01.104

LSL_TCP_USER STRUCT Member ab udVersion 7

Header: In lsl_st_tcp_user.h angegeben

2.7.3.19 OS_TCP_USER_SELECT

Diese Funktion testet die angegebenen Sockets auf Sende- oder Empfangsbereitschaft.

```

VAR_INPUT
  count      : DINT;
  Sockets    : ^DINT;
  select_type: UDINT;
  flags      : UDINT;
  timeout_ms: UDINT;
END_VAR
VAR_OUTPUT
  retval    : DINT;
END_VAR;

```

Übergabeparameter	Typ	Beschreibung
count	DINT	Anzahl der Einträge im Socket-Nummern-Array, auf das psocket zeigt
Sockets	^DINT	Zeiger auf ein Array mit den Nummern jener Sockets, die überprüft werden sollen. Die Socket-Nummern werden von den Funktionen

		OS_TCP_USER_SOCKET() bzw. OS_TCP_USER_SOCKET_EX() zurückgegeben.				
select_type	UDINT	<p>SELECT_OPT_READ: überprüft, ob mindesten ein Socket zum Lesen bereit ist.</p> <p>SELECT_OPT_WRITE: überprüft, ob mindesten ein Socket zum Schreiben bereit ist.</p> <p>SELECT_OPT_EXCEPTION: überprüft, ob bei mindesten einem Socket eine Ausnahmebedingung vorliegt.</p> <p>Diese Werte können nicht miteinander kombiniert werden.</p>				
flags	UDINT	Nicht verwendet, muss auf 0 gesetzt werden				
timeout_ms	UDINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td> <td>Funktion blockiert nicht</td> </tr> <tr> <td>>0</td> <td>Timeout in ms, Funktion blockiert</td> </tr> </table>	0	Funktion blockiert nicht	>0	Timeout in ms, Funktion blockiert
0	Funktion blockiert nicht					
>0	Timeout in ms, Funktion blockiert					
Rückgabeparameter	Typ	Beschreibung				
retval	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">>0</td> <td>Gesamtzahl der bereiten Sockets</td> </tr> <tr> <td><0</td> <td>Fehler-Code</td> </tr> </table> <p>TCP_NOT_READY: die Anzahl der bereiten Sockets ist null. TCP_INVALID_SOCKETNUM: keinen gültigen Socket gefunden. TCP_INVALID_PARAM: der Zeiger sockets ist ungültig oder select_type ist unbekannt.</p>	>0	Gesamtzahl der bereiten Sockets	<0	Fehler-Code
>0	Gesamtzahl der bereiten Sockets					
<0	Fehler-Code					

Diese Funktion testet count Sockets ab sockets auf Sende- oder Empfangsbereitschaft. Der aufrufende Task wird blockiert, bis eine Aktivität auf einem der angegebenen Sockets stattfindet oder bis die Zeit timeout_ms abgelaufen ist.

Die angegebenen Sockets werden laut dem Parameter select_type überprüft, ob sie zum Lesen bereit sind oder ob sie zum Schreiben bereit sind oder ob Ausnahmebedingungen vorliegen.

Ein Server-Socket gilt als bereit zum Lesen, wenn eine Verbindung ansteht, die mit OS_TCP_USER_ACCEPT() angenommen werden kann. Ein Client-Socket ist bereit zum Schreiben, wenn seine Verbindung vollständig aufgebaut ist.

Mit Ausnahmebedingungen sind keine Fehler gemeint. Fehler werden sofort gemeldet, wenn ein fehlerhafter Systemaufruf ausgeführt wird. Vielmehr umfassen sie Bedingungen wie das Vorhandensein einer dringenden Nachricht auf einem Socket.

Der Parameter timeout_ms gibt die maximale Wartezeit in Millisekunden an. Geben Sie Null als Zeit an, wenn Sie herausfinden wollen, welche Sockets bereit sind, ohne zu warten, wenn keine bereit sind.

Der normale Rückgabewert von OS_TCP_USER_SELECT() ist die Gesamtzahl der bereiten Sockets. Wenn die Anzahl der bereiten Sockets null ist, dann wird TCP_NOT_READY zurückgegeben. Welche Sockets für die entsprechende Operation bereit sind, geht aus dem Socket-Nummern-Array hervor. Ist der Inhalt eines Array-Elements ungleich Null, dann ist der Socket bereit. Ist der Inhalt eines Array-Elements gleich Null, dann ist der Socket nicht bereit.

Wenn ein Fehler auftritt, gibt OS_TCP_USER_SELECT() einen negativen Fehlercode zurück. Die Elemente des Socket-Nummern-Arrays sind null.

2.7.3.20 OS_TCP_USER_SEND

Daten an einen Anschluss schicken.

```
VAR_INPUT
  socket      : DINT;
  buffer      : ^CHAR;
  buflen      : UDINT;
  flags       : UDINT;
  timeout_ms : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
socket	DINT	Socket-Nummer	
buffer	^CHAR	Zu sendender Puffer	
buflen	UDINT	Anzahl der zu sendenden Byte	
flags	UDINT	Nicht verwendet, muss auf 0 gesetzt werden	
timeout_ms	UDINT	0 Funktion blockiert nicht >0 Timeout in ms, Funktion blockiert	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	>0 Anzahl von Byte in der Warteschlange zum Senden 0 Ende der Datei <0 Fehler-Code	

Wenn der Remote nicht zum Empfangen der Daten bereit ist, liefert die Funktion TCP_NOT_READY zurück, wenn timeout_ms auf 0 gesetzt ist (muss abgefragt werden).

Ist timeout_ms > 0, wird die Funktion blockiert bis der Remote zum Empfang bereit, ein Fehler auftritt oder das Zeitlimit abgelaufen ist.

Die Funktion wird blockiert, bis alle Daten an den Remote gesendet wurden.

Große Datenmengen sollten durch mehrere Funktionsaufrufe in kleineren Blöcken gesendet werden.

Beispiel

```
retval: DINT;
buffer : ARRAY[0..1024] of CHAR;

retval := OS_TCP_USER_SEND(socket, #buffer[0], sizeof(buffer), 0, 0);
if retval <= 0 & retval >> TCP_NOT_READY then
...error
elsif retval > 0 then
...sent...
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In Isl_st_tcp_user.h angegeben

2.7.3.21 OS_TCP_USER_SENDTO

Daten an einen Anschluss schicken.

```
VAR_INPUT
  socket      : DINT;
  buffer      : ^CHAR;
  buflen      : UDINT;
  flags       : UDINT;
  timeout_ms  : UDINT;
  ipaddr      : UDINT;
  port        : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
socket	DINT	Socket-Nummer
buffer	^CHAR	Zu sendender Puffer

buflen	UDINT	Anzahl der zu sendenden Byte
flags	UDINT	Nicht verwendet, muss auf 0 gesetzt werden
timeout_ms	UDINT	0 Funktion blockiert nicht >0 Timeout in ms, Funktion blockiert
ipaddr	UDINT	Ziel IP-Adresse
port	UDINT	Ziel Port-Nummer
Rückgabeparameter	Typ	Beschreibung
retval	DINT	>0 Anzahl von Byte in der Warteschlange zum Senden <0 Fehler-Code

Die Funktion wirkt wie [OS_TCP_USER_SEND\(\)](#). Der einzige Unterschied ist die Ziel-IP-Adresse und die Port-Nummer, die durch die Parameter ipaddr und port angegeben werden kann. Die Zielparameter sind nur für die UDP-Sockel und wird von TCP-Sockel ignoriert.

Beispiel Senden eines UDP-Datagramms

```

socket  : DINT;
ipaddr  : UDINT;
buffer  : ARRAY [0..63] OF CHAR;

socket := OS_UDP_USER_SOCKET();
if (socket >= 0) then

  rc := 0;

  if (mode = 1) then
    // send an UDP datagram to 200.100.100.110, port 4321 with a
    // random local port value
    ipaddr := OS_TCP_USER_STRTOULONG("200.100.100.110");
  elseif (mode = 2) then
    // send an UDP broadcast message to port 4321 with a random local
    // port value
    // send an UDP broadcast message to port 4321 out on all available
    // ethernet interfaces with a random local port number
    ipaddr := IP_ADDR_BROADCAST;
  else (mode = 3) then
    // send an UDP broadcast message out on an specified interface with
    // a local port = 4322, local IP address = 200.100.100.100
    ipaddr := OS_TCP_USER_STRTOULONG("200.100.100.100");

  rc := OS_UDP_USER_BIND(socket, ipaddr, 4322);
  ipaddr := IP_ADDR_BROADCAST;

```

```

end_if;

if (rc = 0) then

    _strcpy(#buffer[0], "UDP Message Test");

    rc := OS_TCP_USER_SENDTO(socket,
                            #buffer[0],
                            sizeof(buffer),
                            0,
                            2000,
                            ipaddr,
                            4321);

    if (rc < 0) then
        // error ...
    end_if;
end_if;
end_if;

```

Anforderungen

Version: Ab LasalOS 01.01.104

LSL_TCP_USER STRUCT Member ab udVersion 7

Header: In lsl_st_tcp_user.h angegeben

2.7.3.22 OS_TCP_USER_SETSOCKOPT

Eine Sockeloption setzen.

```

VAR_INPUT
    socket      : DINT;
    level       : DINT;
    option_name : DINT;
    option_value : ^CHAR;
    optionlen   : DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;

```

Übergabeparameter	Typ	Beschreibung
socket	DINT	Socket-Nummer
level	DINT	Protokollebene (SOL_SOCKET oder IPPROTO_IP)
option_name	DINT	Zu ändernde Option. Siehe die folgende Tabelle für die unterstützten

option_value	^CHAR	Zeiger auf einen Puffer, der die Werte für die angegebene Option enthält
optionlen	DINT	Länge der Daten, auf den der Parameter option_value zeigt
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich <0 Fehler-Code

Die folgende Tabelle ist eine Zusammenfassung der unterstützten Protokollebenen und Optionen.

level option_name	Standard	option_value	Bedeutung
SOL_SOCKET SO_DELAYED_ACK	Aktiviert	Typ DINT. 0 bedeutet, deaktivieren; Nicht-0 bedeutet wiederum Option aktivieren.	Verzögert das Senden der TCP-Quittierung. In einem Strom von Full-Size-Paketen wird nur jedes zweite Paket quittiert.
SOL_SOCKET SO_NAGLE	Aktiviert	Typ DINT. 0 bedeutet, deaktivieren; Nicht-0 bedeutet wiederum Option aktivieren.	Der Nagel-Algorithmus verbietet das Senden von kleinen TCP-Paketen (weniger als MSS), wenn es noch ausstehende Output-Daten gibt, die nicht quittiert wurden.
SOL_SOCKET SO_BROADCAST	Aktiviert	Typ DINT. 0 bedeutet, deaktivieren; Nicht-0 bedeutet wiederum Option aktivieren.	Diese Option ermöglicht das Versenden von Broadcasts über einen UDP-Socket und wird ab Salamander Version 09.03.060 unterstützt.
SOL_SOCKET SO_REUSEADDR	Deaktiviert	Typ DINT. 0 bedeutet, deaktivieren; Nicht-0 bedeutet wiederum Option aktivieren.	Ermöglicht die Wiederverwendung lokaler Adressen. Wenn eine Serveranwendung eine lokale Adresse an einen Socket bindet und dann beginnt, Verbindungen auf diesem zu akzeptieren (OS_TCP_USER_ACCEPT), wird die lokale Adresse an den lokalen Socket gebunden. Wenn die Server-Anwendung angehalten und neu gestartet wird, schlägt ein anschließender Versuch, die lokale Adresse zu binden, fehl. Dies ist als "address in use"-Fehler bekannt. Der Grund dafür ist, dass sich der Server-Socket im Zustand WAIT_STATE befindet. Zwei Minuten

			lang bleibt der Socket in diesem Zustand und wird dann freigegeben, so dass die lokale Adresse wiederverwendet werden kann. Um die Möglichkeit der Wiederverwendung der lokalen Adresse vor Ablauf der Zwei-Minuten-Frist zu erzwingen, kann die Option SO_REUSEADDR verwendet werden. Um die Wiederverwendung zu ermöglichen, muss die Option vor dem Aufruf von OS_UDP_USER_BIND aktiviert werden. Bei der Verwendung dieser Option ist Vorsicht geboten, da sie TCP weniger zuverlässig macht.
IPPROTO_IP IP_MULTICAST_IF	Nicht gesetzt	Datentyp UDINT. IP-Adresse in Netzwerk-Byte-Reihenfolge. In Bezug auf die Form der übergebenen IP-Adresse gelten die Anmerkungen zum Parameter mc_group_addr der Socket-Option IP_ADD_MEMBERSHIP.	Festlegung der zum Senden von Multicast-IP-Paketen über den betroffenen Socket verwendeten lokalen Ethernet-Schnittstelle. Bitte beachten Sie auch die unten angeführten Hinweise zu Multicast-IP-Adressen und zum Internet Group Management Protocol (IGMP) bzw. IGMP-Snooping.
IPPROTO_IP IP_MULTICAST_LOOP	Aktiviert	Typ DINT. 0 bedeutet, deaktivieren; Nicht-0 bedeutet wiederum Option aktivieren.	Deaktivierung bzw. Aktivierung des Loopbacks von über den Socket gesendeten Multicast-Paketen an die lokalen Sockets. Im Normalfall werden IP-Pakete, die über einen Socket an eine Multicast-Gruppe gesendet werden, auch an Sockets in der eigenen Steuerung weitergegeben bzw. zurückgeleitet, die Mitglieder dieser Multi-cast-Gruppe sind, d.h. auf eingehende, an die Adresse dieser Gruppe gesendete IP-Pakete warten. Die beschriebene Option erlaubt die Deaktivierung dieser Weitergabe und eine daraus resultierende Einsparung von Ressourcen. Diese Maßnahme sollte jedoch mit Vorsicht gesetzt werden, da dadurch der gesamte von einem Socket ausgehende Verkehr zwar an entsprechend konfigurierte Sockets auf anderen Steuerungen bzw. Rechnern weitergeleitet wird, nicht aber an Anwendungen, die auf der eigenen Steuerung ausgeführt werden.

IPPROTO_IP IP_ADD_MEMBERSHIP	Deaktiviert	Datentyp IP_MC_REQ Details siehe unten.	Diese Option wird verwendet, um den Empfang von Multicast-IP-Paketen für eine bestimmte Multicast-IP-Adresse zu ermöglichen. Details siehe unten. Bitte beachten Sie auch die unten angeführten Hinweise zu Multicast-IP-Adressen und zum Internet Group Management Protocol (IGMP) bzw. IGMP-Snooping.
---------------------------------	-------------	-----------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Option IP_ADD_MEMBERSHIP

Diese Option wird verwendet, um den Empfang von Multicast-IP-Paketen für eine bestimmte Multicast-IP-Adresse zu ermöglichen bzw. einer IP-Multicast-Gruppe als Empfänger beizutreten. In option_value wird die Adresse einer IP_MC_REQ-Struktur übergeben. Der Datentyp IP_MC_REQ ist in derselben Header-Datei wie OS_TCP_USER_SETSOCKOPT (isl_st_tcp_user.h) definiert.

```
TYPE
  IP_MC_REQ : STRUCT
    mc_group_addr : UDINT;      // IP address of multicast group
    iface_addr    : UDINT;      // IP address of local interface used to join multicast
  END_STRUCT;
END_TYPE
```

mc_group_addr enthält die Adresse der Multicast-Gruppe, welcher die Anwendung beitreten möchte. Ihr Wert muss daher eine gültige Multicast-Adresse sein und als vorzeichenlose 32-Bit-Ganzzahl übergeben werden. Die Anordnung der Bytes muss der unabhängig von der Hardware für TCP/IP geltenden Netzwerk-Byte-Anordnung (Big-Endian) folgen. Es wird empfohlen, die Methode ConvertStrToBin() der OS-Interface-Klasse _IP zu verwenden, um diesem Parameter ausgehend von einer IP-Adresse in der üblichen „dotted decimal notation“ den richtigen Wert im richtigen Format zuzuweisen. Alternativ kann der zuzuweisende Wert errechnet werden, indem die vier Ganzahlen der IP-Adresse in der genannten Notation mit Potenzen der Basis 256 multipliziert und aufsummiert werden. Auf Little-Endian-Plattformen, z.B. Edge, Edge 2 und x86, wird dazu vor der Summenbildung die erste Ganzzahl mit 1, die zweite mit 256, die dritte mit 256² und die vierte mit 256³ multipliziert. Auf Big-Endian-Plattformen müssten die genannten Faktoren in umgekehrter Reihenfolge angewendet werden.

Sofern der Entwickler die verwendete Multicast-IP-Adresse selbst festlegen muss, sind dabei die entsprechenden Standardisierungsdokumente, vor allem Abschnitt 6 des RFC 2365 (Administratively Scoped IP Multicast) der IETF zu beachten. Einige grundlegende Hinweise zu Multicast-IP-Adressen finden sich auch im Abschnitt „Zur Verwendung von Multicast-IP-Adressen aus dem Administratively Scoped IPv4 Multicast Space“ weiter unten in diesem Dokument.

iface_addr ist die Adresse der lokalen Ethernet-Schnittstelle, über die das System die an die Multicast-Gruppe gesendeten IP-Pakete empfangen wird. Die Form der übergebenen IP-Adresse ist dieselbe wie beim Parameter mc_group_addr. Wenn die Adresse gleich IP_ADDR_ANY (0.0.0.0) ist, versucht das Betriebssystem eine entsprechende Ethernet-Schnittstelle auszuwählen. Es wird jedoch empfohlen, dem Member iface_addr immer die IP-Adresse der gewünschten Ethernet-Schnittstelle zuzuweisen. Dadurch können Fehler (z.B. -4019/No such device) beim Setzen der Option IP_ADD_MEMBERSHIP vermieden werden.

Für den korrekten Empfang der Multicast-Pakete ist außerdem der Aufruf der Funktion OS_UDP_USER_BIND notwendig. Dabei reicht es aus, dem Socket eine Kombination aus IP_ADDR_ANY und der gewünschten Port-Nummer zuzuweisen.

Zur Verwendung von Multicast-IP-Adressen aus dem Administratively Scoped IPv4 Multicast Space

Der Begriff „Administratively Scoped IPv4 Multicast Space“ bezeichnet einen Teilbereich innerhalb des für Multicast bestimmten Abschnitts des IPv4-Adressraums. RFC 2365 der IETF mit dem Titel „Administratively Scoped IP Multicast“ definiert 239.0.0.0 als untere und 239.255.255.255 als obere Grenze dieses Teilbereichs. Wenn keine Multicast-IP-Adresse vorgegeben ist, wird dazu geraten, neben etwaigen anderen Standardisierungsdokumenten die Vorgaben des genannten RFC zu berücksichtigen und eine IP-Adresse aus dem genannten Teilbereich, konkret aus dem „IPv4 Local Scope“, zu wählen. Dieser Abschnitt ist Teil des „Administratively Scoped IPv4 Multicast Space“ und reicht von 239.255.0.0 bis 239.255.255.255.

Eine als Zieladresse eines IP-Pakets übertragene Multicast-IP-Adresse aus dem „IPv4 Local Scope“ identifiziert zusammen mit der als Quelladresse übertragenen IP-Adresse der Sendeschnittstelle eine Gruppe von Knoten im selben IP-Subnetz wie der Multicast-Sender, die die exklusiven Empfänger der Paket-Nutzdaten sind. Zusammen mit der bzw. den an einer IP-Schnittstelle eingestellten Kombinationen von IP-Adressen und Subnetzmasken entscheidet die in einem eingelangten Multicast-IP-Paket enthaltene Quelladresse über dessen weitere Verarbeitung. Ein für den Empfang von Multicast-Paketen mit der betrachteten Zieladresse über die betrachtete Schnittstelle konfigurierter Socket erhält ein Paket nur, wenn dessen Quell-IP-Adresse in einem IP-Subnetz liegt, an dem auch die betrachtete IP-Schnittstelle teilnimmt. An einer Ethernet-Schnittstelle empfangene Ethernet-Frames bzw. darin enthaltene Multicast-IP-Pakete können daher dieselbe Multicast-IP-Adresse als Zieladresse aufweisen, jedoch für unterschiedliche IP-Subnetze bzw. Gruppen von Multicast-Empfängern bestimmt sein. Dies ist dann der Fall, wenn die als Quelladressen in den IP-Paketen angeführten IP-Adressen der Senderschnittstellen in unterschiedlichen IP-Subnetzen liegen.

Beim Setzen der Socket-Option `IP_MULTICAST_IF` in einer Sender-Applikation wird nicht nur die zum Senden von Multicast-IP-Paketen über den betroffenen Socket verwendete lokale Schnittstelle bestimmt, sondern auch die in den Paketen angeführte Quell-IP-Adresse. Im Gegensatz dazu legt die beim Setzen von `IP_ADD_MEMBERSHIP` durch eine Empfänger-Applikation in `iface_addr` übergebene Schnittstellenadresse nur die zum Empfangen von Multicast-IP-Paketen über den betroffenen Socket verwendete lokale Schnittstelle fest. Damit der empfangende Socket Daten aus IP-Paketen mit einer bestimmten Quell-IP-Adresse erhält, muss die in `iface_addr` übergebene Schnittstellenadresse jedoch nicht zwingend im selben IP-Subnetz wie die in den Paketen enthaltene Quell-IP-Adresse liegen. Zwar muss die auf diese Weise spezifizierte Schnittstelle an dem vom Sender mittels `IP_MULTICAST_IF` festgelegten IP-Subnetz teilnehmen, dies kann jedoch auch durch eine weitere, an der Schnittstelle eingestellte IP-Adresse gegeben sein. Multicast-IP-Pakete, deren Zieladresse zwar der in `mc_group_addr` übergebenen Multicast-IP-Adresse entspricht, deren Quelladresse jedoch nicht in einem entsprechenden IP-Subnetz liegt, sind für Mitglieder einer Gruppe von Multicast-Empfängern bestimmt, die zwar durch dieselbe Multicast-IP-Adresse, jedoch in einem anderen IP-Subnetz definiert ist, und werden daher nicht an den Socket weitergegeben, auch wenn sie über die in `iface_addr` spezifizierte Schnittstelle empfangen worden sind.

Wenn eine Sendeschnittstelle zwar am selben IP-Subnetz teilnimmt wie eine Empfangsschnittstelle, jedoch auch noch einen Knoten eines anderen IP-Subnetzes darstellt, d.h. mehrere Kombinationen von IP-Adresse und Subnetzmaske aufweist, besteht somit ein gewisses Risiko, dass ein für den Empfang konfigurierter Multicast-Socket keine Daten eines Sende-Sockets erhält, obwohl die entsprechenden Sockets fehlerfrei für Multicast konfiguriert bzw. die genannten Socket-Optionen ohne Fehler gesetzt werden konnten. In so einem Fall sollte geprüft werden, ob die beim Setzen der Socket-Option `IP_MULTICAST_IF` in der Sende-Applikation übergebene Schnittstellen-IP-Adresse im richtigen IP-Subnetz liegt, da sie auch als Quelladresse in die einzelnen Multicast-IP-Pakete eingetragen wird.

IGMP-Snooping

Dieser Begriff bezeichnet eine Funktion von handelsüblichen, meist verwaltbaren Ethernet-Switches, die dazu dient, die negativen Auswirkungen von Multicast-Übertragungen auf Auslastung und Sicherheit eines Netzwerkes möglichst gering zu halten. Das Internet Group Management Protocol (IGMP) ist ein IPv4-basiertes Protokoll, durch das Router und Hosts innerhalb eines IP-Subnetzes Informationen zu Mitgliedschaften in Multicast-Gruppen austauschen können. IGMP bildet die Basis für Multicast in IPv4-Netzen, indem es eine möglichst zielgerichtete Übertragung von Multicast-Daten an jene Hosts innerhalb des Netzwerkes ermöglicht, die sie benötigen bzw. angefordert haben. Ein sogenannter IGMP-Querier stellt u.a. periodische Anfragen

an alle möglichen Multicast-Empfänger innerhalb eines IP-Subnetzes. Bei diesem IGMP-Querier handelt es sich in der Regel um einen Router, der ggf. aus mehreren im Netz vorhandenen Routern ausgewählt worden ist. Durch die IGMP-Membership-Querys werden ggf. IGMP-Membership-Reports ausgelöst, durch die Hosts im Netz über bestehende Mitgliedschaften in Multicast-Gruppen informieren. Hosts senden solche IGMP-Membership-Reports auch unaufgefordert und ohne regelmäßige Wiederholung, wenn sie Teilnehmer einer Multicast-Gruppe werden wollen.

Beim IGMP-Snooping verarbeitet ein Switch die Daten der von den Hosts eines Netzwerks gesendeten IGMP-Pakete und gelangt so an Informationen über die Mitgliedschaften der Hosts in diversen Multicast-Gruppen. Diese Informationen ermöglichen es dem Switch, an einem Port eingelangte Multicast-IP-Pakete eines Multicast-Senders nur an jene Ports weiterzuleiten, hinter denen sich Hosts befinden, die den Empfang der betreffenden Pakete wünschen. So kann eine unnötige Belastung des Netzes durch Multicast-IP-Pakete vermieden werden. IGMP-Snooping kann jedoch dazu führen, dass die Übertragung von Multicast-Daten zu einem Teil der Empfänger einer Multicast-Gruppe für einen beliebig großen Zeitraum unterbrochen wird. Solche Unterbrechungen drohen in einem IP-Subnetz, in dem kein Router bzw. Multicast-Querier vorhanden ist und somit die Informationen eines Switches über eventuelle Multicast-Empfänger an dessen Ports nicht regelmäßig aktualisiert werden. In einem solchen Fall wird empfohlen, mögliche Auswirkungen auf Performance und Sicherheit des betroffenen Netzwerkes gegen mögliche Auswirkungen von Störungen in der Übertragung von Multicast-Daten abzuwagen und IGMP-Snooping ggf. zu deaktivieren. Von dieser Empfehlung ausgenommen sind Multi-Layer-Switches, die selbst die Fähigkeit zur Durchführung von IGMP-Membership-Querys besitzen, um den Einsatz von IGMP-Snooping auch ohne einen IGMP-fähigen Router im Layer 2-Netz zu ermöglichen.

Beispiel

```
retval  : DINT;
onOff   : DINT;

if OS_TCP_USER_VERSION >= 5 then
  onOff := 0;
  retval := OS_TCP_USER_SETSOCKOPT(newsock,
                                    SOL_SOCKET,
                                    SO_DELAYED_ACK,
                                    #onOff$CHAR,
                                    sizeof(onOff));
else
  retval := -1;
end_if;

if retval < 0 then
  ...error
end_if;
```

Anforderungen

Version: Ab LasalOS 01.01.051

LSL_TCP_USER STRUCT Member ab udVersion 5

Header: In lsl_st_tcp_user.h angegeben

2.7.3.23 OS_TCP_USER_SHUTDOWN

Deaktiviert das Empfangen bzw. das Senden an einen Sockel.

```
VAR_INPUT
    socket      : DINT;
    how        : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
socket		DINT	Socket-Nummer
how		UDINT	0 Empfangen deaktivieren 1 Senden deaktivieren 2 Beides deaktivieren
Rückgabeparameter		Typ	Beschreibung
retval		DINT	0 Funktion erfolgreich <0 Fehler-Code

Die Shutdown-Funktion schließt den Sockel nicht. Alle Ressourcen, die mit dem Sockel verbunden sind, werden nicht freigegeben, bis die Schließen-Sockel-Funktion aufgerufen ist.

Um sicherzustellen, dass alle Daten über einen Sockel gesendet und empfangen werden, bevor dieser geschlossen wird, sollte die Abschaltfunktion zunächst verwendet werden, um die Verbindung vor dem Aufruf von closesocket zu schließen

Beispiel

```
retval : DINT;

retval := OS_TCP_USER_SHUTDOWN(socket, 2);
if retval < 0 then
```

```

    ...error
else
    ...shutdown...
end_if;

```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In Isl_st_tcp_user.h angegeben

2.7.3.24 OS_TCP_USER_SOCKET

Einen TCP-Socket reservieren.

```

VAR_OUTPUT
    retval      : DINT;
END_VAR;

```

Rückgabeparameter	Typ	Beschreibung	
retval	DINT	≥0	Socket-Nummer <0 Fehler-Code

Wenn mehrere Ethernet-Schnittstellen verfügbar sind, können über alle Daten ausgetauscht werden.

Wird [OS_TCP_USER_CONNECT\(\)](#) verwendet, um einen Remote-Host zu verbinden, entscheidet der IP-Stack des Betriebssystems über welche Ethernet-Schnittstelle das Paket geschickt wird.

Bei der Verwendung von [OS_TCP_USER_LISTEN\(\)](#), wo auf eingehende Verbindungen eines Remote Host gewartet wird, ist der geöffnete Port für alle Schnittstellen gültig.

Diese Funktion ist in Betriebssystem-Versionen ab 01.01.100 vorhanden.

Sollen Daten über eine bestimmte Schnittstelle gesendet werden oder eingehende Verbindungen nur für eine bestimmte Schnittstelle akzeptiert werden, kann [OS_TCP_USER_SOCKET_EX\(\)](#) verwendet werden.

Beispiel

```

socket  : DINT;

socket := OS_TCP_USER_SOCKET( );
if socket < 0 then
    ...error

```

```
else
    ...o.k.
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.25 OS_TCP_USER_SOCKET_EX

Einen TCP-Socket reservieren.

```
VAR_INPUT
    iface      : DINT;
END_VAR
VAR_OUTPUT
    retval     : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
iface		DINT	Netzwerkschnittstelle zur Aufteilung der Sockel auf 1 für die erste Schnittstelle, 2 für die zweite ...
Rückgabeparameter		Typ	Beschreibung
retval		DINT	≥ 0 Socket-Nummer < 0 Fehler-Code

Beispiel

```
socket  : DINT;

socket := OS_TCP_USER_SOCKET_EX(2); // second interface
if socket < 0 then
    ...error
else
    ...o.k.
end_if;
```

Anforderungen

Version: Ab LasalOS 5.52

LSL_TCP_USER STRUCT Member ab udVersion 2

Header: In Isl_st_tcp_user.h angegeben

2.7.3.26 OS_TCP_USER_STRTOULONG

Wandelt eine Punkt-dezimal-Format IP-Adresse in eine unsigned long IP-Adresse um.

Wenn in der Datei autoexec.lsl ein DNS-Server konfiguriert ist, kann statt der IP-Adresse auch der DNS-Name als CHAR-String angegeben werden (DNS-Lookup). Dieser wird dann aufgelöst und die IP-Adresse wird ins 32-Bit-Binär-Format umgewandelt.

```
VAR_INPUT
    buffer      : ^CHAR;
END_VAR
VAR_OUTPUT
    retval     : UDINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
buffer	^CHAR	Puffer, der die IP-Adresse enthält
Rückgabeparameter	Typ	Beschreibung
retval	UDINT	IP-Adresse 0xFFFFFFFF, wenn ein Fehler festgestellt wird

Beispiel

```
ipaddr  : UDINT;
ipstr   : ARRAY[0..15] of CHAR;

_strcpy(#ipstr[0], "192.168.44.105");

ipaddr := OS_TCP_USER_STRTOULONG(#ipstr[0]);
if ipaddr = 0xFFFFFFFF then
...error
  else
...o.k.
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In Isl_st_tcp_user.h angegeben

2.7.3.27 OS_TCP_USER_TOIP

Wandelt die IP-Adresse aus 4 INTs in eine Punkt-dezimal-Format IP-Adresse.

```

VAR_INPUT
  buffer      : ^CHAR;
  buflen      : UDINT;
  ID1         : UDINT;
  ID2         : UDINT;
  ID3         : UDINT;
  ID4         : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;

```

Übergabeparameter		Typ	Beschreibung
buffer		^CHAR	Zeiger auf einen Puffer, der die IP-Adresse erhält
buflen		UDINT	Länge des Puffers
ID1		UDINT	Erste ID der IP-Adresse
ID2		UDINT	Zweite ID der IP-Adresse
ID3		UDINT	Dritte ID der IP-Adresse
ID4		UDINT	Vierte ID der IP-Adresse
Rückgabeparameter		Typ	Beschreibung
retval		DINT	0 Funktion erfolgreich <0 Fehler-Code

Beispiel

```

retval  : DINT;
buffer  : ARRAY[0..15] of CHAR;

retval := OS_TCP_USER_TOIP(#buffer[0], sizeof(buffer), 192, 168, 44, 105);
if retval < 0 then
  ...error
else
  ...o.k.
end_if;

```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.28 OS_TCP_USER ULONGTOSTR

Wandelt eine unsigned long IP-Adresse in eine Punkt-dezimal-Format Adresse um.

```
VAR_INPUT
    buffer      : ^CHAR;
    buflen      : UDINT;
    IPAddr      : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
buffer	^CHAR	Zeiger auf einen Puffer, der die IP-Adresse erhält	
buflen	UDINT	Länge des Puffers	
IPAddr	UDINT	IP-Adresse im unsigned long-Format	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	Funktion erfolgreich
		<0	Fehler-Code

Beispiel

```
retval  : DINT;
ipaddr  : UDINT;
ipstr   : ARRAY[0..15] of CHAR;

retval := OS_TCP_USER ULONGTOSTR(#ipstr[0], sizeof(ipstr), ipaddr);
if retval < 0 then
    ...error
else
    ...o.k.
end_if;
```

Anforderungen

Version: Ab LasalOS 5.28

LSL_TCP_USER STRUCT Member ab udVersion 1

Header: In lsl_st_tcp_user.h angegeben

2.7.3.29 OS_TCP_USER_STRTOLONG_ASY

Diese Funktion wird verwendet, um einen Namen in eine unsigned long IP-Adresse zu wandeln. Sie arbeitet asynchron und liefert das Ergebnis über die Callback-Funktion pUserFct zurück.

```
VAR_INPUT
  Buffer      : ^CHAR;
  pUserFct    : ^VOID;
  pUserParam  : ^VOID;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
Buffer		^CHAR	Adresse eines Puffers mit der Pool-Adresse oder IP-Adresse in Dezimalpunktschreibweise (z.B.: us.pool.ntp.org)
pUserFct		^VOID	Adresse der Callback-Funktion zur Rückgabe der aufgelösten IP-Adresse und eines Ergebniscodes (0=Ok, -10 Error)
pUserParam		^VOID	Zeiger auf beliebige Anwenderdaten (z.B. der THIS-Zeiger). Dieser Zeiger wird an die Callback-Funktion übergeben
Rückgabeparameter		Typ	Beschreibung
retval		DINT	0 Kein Fehler <0 Fehler-Code -1 Zeiger Buffer oder Zeiger pUserFct oder Zeiger pUserParam ungültig

Beispiel für eine korrekte Deklaration der Callback-Funktion

```
FUNCTION GLOBAL __CDECL StrToUlongCallback
VAR_INPUT
  pUserParam  : pVoid;      // Zeiger auf beliebige Anwenderdaten (z.B. der this-Pointer)
  IPAddress  : UDINT;      // aufgelöste IP-Adresse
  retCode     : DINT;       // Ergebniscodes (0=Ok, -10 Error)
END_VAR
```

2.7.3.30 OS_UDP_USER_BIND

Bindet einen Sockel mit einer lokalen Port-Nummer und einer IP-Adresse.

```
VAR_INPUT
  socket      : DINT;
  ipaddr      : UDINT;
  port        : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
socket	DINT	Socket-Nummer	
ipaddr	UDINT	Lokale IP-Adresse, die an den angegebenen Sockel gebunden werden soll	
port	UDINT	Lokale Port-Nummer, die verbunden werden soll (0 = zufällige Port-Nummer) vom IP-Stack des Betriebssystems ausgewählt	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	Funktion erfolgreich
		<0	Fehler-Code

Wenn die Funktion mit der IP-Adresse IP_ADDR_ANY und einer bestimmten Port-Nummer verwendet wird, werden alle an die Port-Nummer gesendeten Pakete empfangen.

Wenn IP_ADDR_BROADCAST verwendet wird, werden nur Broadcast-Pakete (IP-Adresse = 255.255.255.255) empfangen.

Wenn die aktuell eingestellte IP-Adresse verwendet wird, werden nur Pakete empfangen, die an die angegebene Adresse (IP und Port-Nummer) adressiert sind.

OS_UDP_USER_BIND muss aufgerufen werden, um UDP-Datagramme zu empfangen. Wird beim Versenden von UDP-Datagrammen OS_UDP_USER_BIND verwendet und für die IP-Adresse IP_ADDR_ANY angegeben, wird das Paket auf der entsprechenden Ethernet-Schnittstelle versendet, wenn mehr als eine Ethernet-Schnittstelle vorhanden ist. In diesem Fall entscheidet der IP-Stack des Betriebssystems, auf welcher Ethernet-Schnittstelle das Paket versendet werden muss (abhängig von der Konfiguration der IP-Adressen). Wird IP_ADDR_BROADCAST verwendet, wird das Paket auf jeder Ethernet-Schnittstelle ausgesendet.

Mehr Informationen finden Sie in den Beispielen der Funktionen [OS_TCP_USER_RECVFROM\(\)](#) und [OS_TCP_USER_SENDTO\(\)](#).



Ab LasalOS 09.01.001

Mit [IP_ADDR_ANY](#) werden, wie mit [IP_ADDR_BROADCAST](#), nur mehr Broadcast-Pakete empfangen. Um normale UDP-Datagramme zu empfangen, muss der Socket an eine gültige IP-Adresse gebunden werden.

Anforderungen

Ab LasalOS 01.01.104.

Ab LSL_TCP_USER STRUCT Mitglied udVersion 7

Header: In lsl_st_tcp_user.h angegeben

2.7.3.31 OS_UDP_USER_SOCKET

Weist einen UDP-Socket an.

```
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Rückgabeparameter	Typ	Beschreibung
retval	DINT	≥ 0 Socket-Nummer < 0 Fehler-Code

Wenn der UDP-Socket nicht mehr benötigt wird, sollte [OS_TCP_USER_CLOSESOCKET\(\)](#) aufgerufen werden, damit dieser wieder für andere Aufrufe in [OS_UDP_USER_SOCKET](#) verfügbar ist. Ein Maximum von 6 UDP-Sockets kann gleichzeitig geöffnet werden.

Die [OS_UDP_USER_BIND\(\)](#) Funktion kann verwendet werden, um die Sockets mit einer lokalen Port-Nummer und der IP-Adresse zu verbinden. Für weitere Details siehe die Beschreibung der Funktion [OS_UDP_USER_BIND\(\)](#).

Für das Senden und Empfangen von UDP-Datagrammen können die Funktionen [OS_TCP_USER_SENDTO\(\)](#) und [OS_TCP_USER_RECVFROM\(\)](#) verwendet werden

Beispiel

```
socket  : DINT;

socket := OS_UDP_USER_SOCKET();
if socket < 0 then
```

```
    ...error
else
    ...o.k.
end_if;
```

Anforderungen

Ab LasalOS 01.01.104.

LSL_TCP_USER STRUCT Member ab udVersion 7

Header: In Isl_st_tcp_user.h angegeben

2.7.4 Defines

Defines für OS_TCP_USER_IPINFO

2.7.4.1 **#define IP_ADDR_ANY**

Beliebige Adresse ([OS_UDP_USER_BIND\(\)](#))

2.7.4.2 **#define IP_ADDR_BROADCAST**

Broadcast-Adresse ([OS_UDP_USER_BIND\(\)](#))

2.7.4.3 **#define IP_OPT_ADDR**

Zum Erhalt der aktuellen IP-Adresse.

2.7.4.4 **#define IP_OPT_ETHERNET_ADDR**

Zum Erhalten der Ethernet-Adresse.

2.7.4.5 **#define IP_OPT_SUBNETMASK**

Zum Erhalt der aktuellen Subnetmaske.

2.7.5 Aufzählung von Ethernet-Schnittstellen

Da eine SPS mehrere Ethernet-Schnittstellen unterstützen kann, wurde vereinbart, diese nach einem festen Schema zu beziffern.

- | | |
|-----|--------------------------------------------------------------|
| 1 | Erste interne Schnittstelle (ETH1) |
| 2 | Ethernet-Schnittstelle bei der EWP (oder ETH2 in der CCL912) |
| 3 | VARAN-Schnittstelle |
| 4-6 | reserviert |
| 7 | USB zum Ethernet-Adapter |
| 8 | Zweite USB zum Ethernet-Adapter |

2.7.6 Beispiele

Einfacher TCP-Echo-Server

```

VAR_PRIVATE
    lsl_tcp_user    : ^LSL_TCP_USER;
    tcp_step        : USINT;
    mainsock        : DINT;
    newsock         : DINT;
    retval          : DINT;
    data            : ARRAY[0..1024] of CHAR;
END_VAR

FUNCTION VIRTUAL GLOBAL TcpServer::Init
    tcp_step := 0;

    if OS_CILGET("TCP_USER", #lsl_tcp_user) then
        lsl_tcp_user := NIL;
    end_if;

END_FUNCTION //VIRTUAL GLOBAL TcpServer::Init

FUNCTION VIRTUAL GLOBAL TcpServer::CyWork
VAR_INPUT
    EAX      : UDINT;
END_VAR
VAR_OUTPUT
    state    : UDINT;
END_VAR

    CASE tcp_step OF
        0:      mainsock := OS_TCP_USER_SOCKET();

```

```
if mainsock < 0 then
    tcp_step := 7;
else
    tcp_step += 1;
end_if;

1:    retval := OS_TCP_USER_LISTEN(mainsock, 50, 10);
if retval < 0 then
    tcp_step := 6;
else
    tcp_step += 1;
end_if;

2:    newsock := OS_TCP_USER_ACCEPT(mainsock, 0);
if newsock < 0 & newsock <> TCP_NOT_READY then
    tcp_step := 6;
elsif newsock >= 0 then
    tcp_step += 1;
end_if;

3:    retval := OS_TCP_USER_RECV(newsock, #data[0], sizeof(data), 0, 0);
if retval <= 0 & retval <> TCP_NOT_READY then
    tcp_step := 5;
elsif retval > 0 then
    tcp_step += 1;
end_if;

4:    retval := OS_TCP_USER_SEND(newsock, #data[0], sizeof(data), 0, 0);
if retval <= 0 & retval <> TCP_NOT_READY then
    tcp_step := 5;
else
    tcp_step -= 1;
    _memset(#data, 0, sizeof(data));
end_if;

5:    OS_TCP_USER_CLOSESOCKET(newsock, 0);
newsock := 0;
tcp_step += 1;

6:    OS_TCP_USER_CLOSESOCKET(mainsock, 0);
mainsock := 0;
tcp_step += 1;

7:
END_CASE;

state:= READY;
END_FUNCTION //VIRTUAL GLOBAL TcpServer::CyWork
```

Simple TCP Client

```
VAR_PRIVATE
    lsl_tcp_user      : ^LSL_TCP_USER;
    tcp_step         : USINT;
```

```

sock          : DINT;
senddata      : ARRAY[0..1024] of CHAR;
recvdata      : ARRAY[0..1024] of CHAR;
retval        : DINT;
END_VAR

FUNCTION VIRTUAL GLOBAL TcpClient::Init
  tcp_step := 0;

  _memset(#recvdata, 0, sizeof(recvdata));
  _memset(#senddata, 0, sizeof(senddata));
  _strcpy(#senddata[0], "client to server test message...");

  if OS_CILGET("TCP_USER", #lsl_tcp_user) then
    lsl_tcp_user := NIL;
  end_if;

END_FUNCTION //VIRTUAL GLOBAL TcpClient::Init

FUNCTION VIRTUAL GLOBAL TcpClient::CyWork
  VAR_INPUT
    EAX      : UDINT;
  END_VAR
  VAR_OUTPUT
    state    : UDINT;
  END_VAR

  CASE tcp_step OF
    0:      sock := OS_TCP_USER_SOCKET();
            if sock < 0 then
              tcp_step := 5;
            else
              tcp_step += 1;
            end_if;

    1:      retval := OS_TCP_USER_CONNECT(sock, 1000, "192.168.44.215", 21, 0);
            if retval < 0 & retval >> TCP_NOT_READY then
              tcp_step := 4;
            elseif retval = 0 then
              tcp_step += 1;
            end_if;

    2:      retval := OS_TCP_USER_SEND(sock, #senddata[0], sizeof(senddata), 0, 0);
            if retval <= 0 & retval >> TCP_NOT_READY then
              tcp_step := 4;
            elseif retval > 0 then
              tcp_step += 1;
            end_if;

    3:      retval := OS_TCP_USER_RECV(sock, #recvdata[0], sizeof(recvdata), 0, 0);
            if retval <= 0 & retval >> TCP_NOT_READY then
              tcp_step := 4;
            elseif retval > 0 then
  
```

```
        tcp_step += 1;
    end_if;

4:      OS_TCP_USER_CLOSESOCKET(sock, 0);
    sock := 0;
    tcp_step += 1;

5:
END_CASE;

state:= READY;
END_FUNCTION //VIRTUAL GLOBAL TcpClient::CyWork
```

2.8 API XTimer

2.8.1 Allgemein

Der XTimer ist ein hochauflösender Timer mit einem 1 μ s Takt. Dieser Timer wird in der Xilinx-FPGA realisiert und ist nur am C-IPC mit Xilinx Plattform ab Flash-Version F5 verfügbar.

2.8.2 Interface-Funktionen XTIMER

Header Dateien: lsl_st_xtimer.h

Dieser Abschnitt beschreibt die Timer-Schnittstelle. Um diese Schnittstelle zu verwenden, muss zuerst ein Zeiger über OS_CILGET Daten erhalten. Der Name der Schnittstelle ist IXTIMER. Wenn der gleiche Name verwendet wird wie im folgenden Beispiel, können die vordefinierten Makros zur Verringerung und Vereinfachung der Programmierung verwendet werden.

2.8.2.1 XtimerInit

Initialisiert den Timer mit einem Intervall und einer Callback-Funktion. Diese Funktion kann jederzeit zur Änderung der Werte aufgerufen werden. [XTimerStop\(\)](#) wird automatisch ausgeführt, bevor die Einstellungen geändert wurden.

Für eine Definition der Callback-Funktion siehe folgendes Beispiel. Das Register esi wird mit dem Wert des Parameters pThis beim Eintragen geladen. Register eax wird mit einem Status-Code geladen. Ist eax beim Eintragen 0 ist alles in Ordnung. Wenn ein Überlauf aufgetreten ist, also mindestens eine Interrupt-Anforderung verloren wurde, wird Bit 1 von eax gesetzt. Dies kann vorkommen, wenn die Interrupt-Routine sehr langwierig ist, das Timer-Intervall sehr gering oder die Maschine sehr überlastet ist.

```
int XTIMER_Init (
```

```

unsigned long  ulNum,
unsigned long  ulValue,
void          *pThis,
void          *pCallback
);

FUNCTION __CDECL GLOBAL P_XTimerInit
VAR_INPUT
    ulNum      : UDINT;
    ulVal      : UDINT;
    pThis      : ^VOID;
    pCallback  : ^VOID;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;

#define SXTIMER_XTMRINIT(p1,p2,p3,p4)
    pSXTimerInfo^.XTimerInit
    $ P_XTimerInit(p1,p2,p3,p4)

```

Übergabeparameter	Typ	Beschreibung	
ulNum	UDINT	Zählerindex	
ulVal	UDINT	Zählerwert in μ s. Der Wertebereich reicht von 1 bis 65535, d.h. ein Intervall von 1 μ s bis ca. 65 ms.	
pThis	^VOID	THIS-Zeiger eines Objekts	
pCallback	^VOID	Zeiger auf eine Callback-Funktion	
Übergabeparameter	Typ	Beschreibung	
retval	DINT	≠0 Funktion erfolgreich 0 Fehler	

Keine der Parameter kann 0 (oder NULL/NIL) sein. Nachdem dieser Befehl durchgeführt wurde, wird der Timer gestoppt.

Folgende Punkte sind zu beachten, damit Zeitprobleme vermieden werden:

- Ein Abstand von nur wenigen Mikrosekunden wird zu einer hohen Systembelastung führen.
- Eine lange Interrupt Routine wird die Latenz erhöhen und den einwandfreien Betrieb des Systems verhindern.
- Eine Endlosschleife oder ähnliches wird das System blockieren.
- Die CLI oder ähnliche Befehle werden nicht empfohlen.
- Diese Funktion muss sorgfältig entwickelt werden.

Die Messung von minimalen Abständen führt zu den folgenden Ergebnissen:

- CIPC 800 Mhz: 15 μ s
- CIPC 400 Mhz: 20 μ s

Das DUT ist ein einfaches Programm, welches die Callback-Funktion in LASAL CLASS durchführt, online über Ethernet war und die Interrupts gezählt hat.

Anforderungen

Version: Ab LasalOS 5.43.

Beispiel

```
#include <lsl_st_xtimer.h>

VAR_GLOBAL
    pSXTimerInfo  : ^LSL_SXTIMER;
    iCnt          : UDINT;
END_VAR

FUNCTION __CDECL GLOBAL MyISR
    IF EAX <> 0 THEN
        // Problems, probably we've lost at least one request!
    END_IF;
    iCnt := iCnt + 1;
END_FUNCTION

FUNCTION VIRTUAL GLOBAL DoNothing::Init
    IF OS_CILGET("IXTIMER", #pSXTimerInfo) THEN
        ELSE
    SXTIMER_XTIMERINIT( 0, 50000, this, #MyISR() );
    END_IF
END_FUNCTION //VIRTUAL GLOBAL DoNothing::Init
```

2.8.2.2 XTimerSetInterval

Initialisiert den Timer mit einem neuen Intervall-Wert. Diese Funktion kann jeder Zeit zur Änderung der Zählerwerte aufgerufen werden. [XTimerStop\(\)](#) wird automatisch ausgeführt, bevor die Einstellungen geändert wurden.

```
int XTimerSetInterval(
    unsigned long    ulNum,
    unsigned long    ulValue
);
FUNCTION __CDECL GLOBAL P_XTimerSetInterval
VAR_INPUT
    ulNum      : UDINT;
    ulVal      : UDINT;
END_VAR
VAR_OUTPUT
```

```

    retval  : DINT;
END_VAR;

#define SXTIMER_XTIMERSETINTERVAL(p1,p2)
  pSXTimerInfo^.XTimer XTimerSetInterval
  $ P_XTimer XTimerSetInterval (p1,p2)

```

Übergabeparameter		Typ	Beschreibung	
ulNum		UDINT	Zählerindex	
ulVal		UDINT	Zählerwert in μ s; der Wertebereich reicht von 1 bis 65536, d.h. ein Intervall von 1 μ s ms bis ca. 65 ms.	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	#0 Funktion erfolgreich 0 Fehler	

Siehe [XTimerInit\(\)](#).

Anforderungen

Version: Ab LasalOS 5.43

Beispiel

```

// change to 100 $\mu$ s
SXTIMER_XTIMERSETINTERVAL( 0, 100 );

```

2.8.2.3 XTimerSetMode

Legt den Timer-Modus fest.

Mit dieser Funktion kann definiert werden, ob der Timer nach jedem Überlauf automatisch neu gestartet werden, oder ob nur ein Überlauf erfolgen soll.

Wird die Funktion nicht aufgerufen, so wird defaultmäßig der Timer nach jedem Überlauf neu gestartet.

```

int XTimerSetMode(
  unsigned long  ulNum,
  unsigned char  usMode
);

FUNCTION __CDECL GLOBAL P_XTimerSetMode
VAR_INPUT
  ulNum      : UDINT;
  usMode     : USINT;

```

```

END_VAR
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SXTIMER_XTIMERSETMODE(p1,p2)
    PSXTimerInfo^.XTimerSetMode
    $ P_XTimerSetMode (p1,p2)

```

Übergabeparameter		Typ	Beschreibung	
ulNum		UDINT	Timer-Modus	
usMode		USINT	SXTIMER_SINGLE_RUN: einmalige Durchführung; der Timer läuft nach dem Starten nur einmal über und wird anschließend nicht mehr automatisch neu gestartet. SXTIMER_CONT_RUN: wiederholte Durchführung der Timer wird nach jedem Überlauf neu gestartet.	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	≠0 Funktion erfolgreich 0 Fehler	

Der Timer-Modus kann einmalig, oder bei Bedarf auch vor dem Start des Timers eingestellt werden.

Die Änderung des Timer-Modus hat keine Auswirkung auf einen bereits gestarteten Timer. Durch den Aufruf von [XTimerInit](#) wird der Timer-Modus wieder auf den Default-Wert SXTIMER_CONT_RUN zurückgesetzt.

Anforderungen

Ab LasalOS 01.02.120, ab Interface-Version 01.01.001

Beispiel

```

// set timer mode to single run
SXTIMER_XTIMERSETMODE( 0, SXTIMER_SINGLE_RUN );

```

2.8.2.4 XTimerStopAllUser

Stoppt alle Timer.

```

void XTimerStopAllUser(
    void
);
FUNCTION __CDECL GLOBAL P_XTimerStopAllUser

```

```

VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR;

#define SXTIMER_XTIMERSTOPALLUSER()
  pSXTimerInfo^.XtimerStopAllUser
  $ P_XTimerStopAllUser()

```

Anforderungen

Version: Ab LasalOS 5.43

Beispiel

```

FUNCTION VIRTUAL GLOBAL DoNothing::StopAllTimer
  SXTIMER_XTIMERSTOPALLUSER(0);
END_FUNCTION //VIRTUAL GLOBAL DoNothing:: StopAllTimer

```

2.8.2.5 XTimerValue

Gibt den aktuellen Wert zurück.

```

void XTimerValue(
  unsigned long ulNum,
);
FUNCTION __CDECL GLOBAL P_XTimerValue
VAR_INPUT
  ulNum      : UDINT;
END_VAR
VAR_OUTPUT
  UlValue    : UDINT
END_VAR;

#define SXTIMER_XTIMERVALUE(p1)
  pSXTimerInfo^.XtimerValue
  $ P_XTimerValue(p1)

```

Übergabeparameter	Typ	Beschreibung
ulNum	UDINT	Zählerindex
Rückgabeparameter	Typ	Beschreibung
UlValue	UDINT	Gibt den aktuellen Wert zurück

Anforderungen

Version: Ab LasalOS 5.43

Beispiel

```

VAR
    udT : UDINT
END_VAR

udT := SXTIMER_XTIMERVALUE(0);
.

. func();

udT := SXTIMER_XTIMERVALUE(0) - udT
// spent udT µs in function func()

```

2.8.2.6 XtimerReset

Timer startet bei Wert 0 neu und zählt weiter.

```

void XTimerReset(
    unsigned long    ulNum,
);
FUNCTION __CDECL GLOBAL P_XTimerReset
VAR_INPUT
    ulNum      : UDINT;
END_VAR
VAR_OUTPUT
END_VAR;

#define SXTIMER_XTIMERRESET(p1)
    pSXTimerInfo^.XtimerReset
    $ P_XTimerReset(p1)

```

Übergabeparameter	Typ	Beschreibung
ulNum	UDINT	Zählerindex

Anforderungen

Version: LasalOS 5.43 or later.

Beispiel

```

FUNCTION VIRTUAL GLOBAL DoNothing::StopReset
    SXTIMER_XTIMERRESET(0);
END_FUNCTION //VIRTUAL GLOBAL DoNothing:: StopReset

```

2.8.2.7 XtimerStop

Stoppt den Timer und deaktiviert den Interrupt.

```
void XTimerStop(
    unsigned long    ulNum,
);
FUNCTION __CDECL GLOBAL P_XTimerStop
VAR_INPUT
    ulNum      : UDINT;
END_VAR
VAR_OUTPUT
END_VAR;

#define SXTIMER_XTIMERSTOP(p1)
    psXTimerInfo^.XTimerStop
    $ P_XTimerStop(p1)
```

Übergabeparameter	Typ	Beschreibung
ulNum	UDINT	Zählerindex

Anforderungen

Version: Ab LasalOS 5.43

Beispiel

```
FUNCTION VIRTUAL GLOBAL DoNothing::StopTimer
    SXTIMER_XTIMERSTOP(0);
END_FUNCTION //VIRTUAL GLOBAL DoNothing:: StopTimer
```

2.8.2.8 XTimerStart

Startet den Timer und aktiviert den Interrupt.

```
void XTimerStart(
    unsigned long ulNum,
);
FUNCTION __CDECL GLOBAL P_XTimerStart
VAR_INPUT
    ulNum      : UDINT;
END_VAR
VAR_OUTPUT
END_VAR;

#define SXTIMER_XTIMERSTART(p1)
    psXTimerInfo^.XTimerStart
    $ P_XTimerStart(p1)
```

Übergabeparameter	Typ	Beschreibung
ulNum	UDINT	Zählerindex

Anforderungen

Version: Ab LasalOS 5.43

Beispiel

```
FUNCTION VIRTUAL GLOBAL DoNothing::StartTimer
  ICnt := 0;
  SXTIMER_XTIMERSTART(0);
END_FUNCTION //VIRTUAL GLOBAL DoNothing:: StartTimer
```

2.9 Web Server

2.9.1 Befehl

Der LasalOS Web-Server ist eine einfache Web-Server Applikation, die benutzerdefinierte Webseiten in die HTML-Sprache interpretiert. LasalOS kommuniziert mit einem Remote Web-Browser über TCP mittels HTTP-Protokoll.

2.9.2 Anforderungen

LasalOS	Ab Version 01.01.50, websvr.dlm Version 01.01.002 erforderlich Ab Version 01.01.103 für DTC081-IP Keine websvr.dlm ist für DTC081-IP notwendig
Platform	Alle Plattformen mit einer Ethernet-Schnittstelle

2.9.3 Installation

Die WEBSVR.DLM muss in das Verzeichnis C:\LSLSYS kopiert werden.

Um zusätzliche Funktionen innerhalb von LasalOS Web-Seiten verwenden zu können, muss WEBFNC.DLM auch in das Verzeichnis C:\LSLSYS

kopiert werden (siehe Kapitel [Zusätzliche Funktionen](#) für weitere Informationen).

Das Standardverzeichnis des Web-Servers ist C:\

Um dieses Verzeichnis zu ändern, muss die Umgebungsvariable Webroot vor dem Start des Web-Servers gesetzt werden.

Der SETENV Webroot Befehl kann im Command Line Interface verwendet werden bzw. die Zeile kann in der autoexec.lsl hinzugefügt werden, um beim Start des Systems den Web-Server automatisch zu starten.

Das Standardverzeichnis benötigt einen Backslash.

z.B.: SETENV WEBROOT C:\WEBROOT

Das Verzeichnis C:\LSLSYS\HTML ist für LasalOS Web-Seiten reserviert.



Der LasalOS Web-Server kann im CLI mit WEBSVR START gestartet werden oder man bearbeitet die autoexec.lsl um den Server automatisch beim Hochfahren zu starten.

Ist der Web-Server gestartet, wird die Web-Server-DLL geladen. Wird der Web-Server nicht mehr benötigt, kann die DLL mit dem Befehl WEBSVR STOP gelöscht werden.

Im Kapitel Command Line Interface (CLI) Kommandos findet man eine detaillierte Beschreibung der einzelnen CLI-Befehle.

Falls Sicherheit notwendig ist, sind zwei Setup-Befehle verfügbar.

WEBSVR FILTER +Adresse / Maske,

um einen Host / Netzwerk-Eintrag einzufügen, die der Web-Server mit ankommenden Verbindungen vergleicht.

WEBSVR AUTH ADD webpage realm username: Passwort

um einen Eintrag für Benutzeridentifikation für das ganze Hauptverzeichnis oder mehrere Web-Seiten einzufügen.

Die WEBSVR AUTH Befehl kann nicht innerhalb der autoexec.lsl verwendet werden.

Siehe Kapitel [Sicherheit](#) für mehr Informationen über die Authentifizierung!

2.9.4 Sicherheit

2.9.4.1 IP-Adresse Authentifizierung

Mit dem LasalOS CLI-Befehl [WEBSVR FILTER](#) oder einem API-Aufruf aus der Anwendung kann eine Liste der zulässigen IP-Adressen eingerichtet werden. Wenn kein Filter gesetzt ist, akzeptiert der Server alle IP-Adressen, die durch die Netzwerkkonfiguration erreicht werden können. Wenn ein einzelner Filter gesetzt ist, akzeptiert der Web-Server nur Verbindungen, die durch die Filtereinstellung erlaubt sind. Es kann mehr als ein Filter gesetzt werden (max. 5, Web-Server-DLL ab Version 01.01.001).

Zwei verschiedene Filtertypen können eingegeben werden:

Host-Einträge:	Der Server akzeptiert nur Verbindungen von dieser IP-Adresse. Die Maske ist daher immer 255.255.255.255.
NETWORK Einträge:	Der Server akzeptiert alle Verbindungen, die innerhalb des Netzwerks existieren. Mit dieser Einstellung wird die Maske von 255.255.255.255 variiert.

Beispiel

Host Eintrag: 10.100.100.100 / 255.255.255.255
10.100.100.101 / 255.255.255.255

Der Server akzeptiert nur eine Verbindung vom Remote-Host 10.100.100.100 und 10.100.100.101

Netzwerkeintrag: 10.100.100.0 / 255.255.255.0

Der Server akzeptiert Verbindungen vom Remote-Host 10.100.100.X

Sind beide Host- und Netzwerk-Einträge gesetzt und der Host-Eintrag ist im selben Netzwerk wie der Netzwerk-Eintrag, wird der Host-Eintrag ignoriert.

Beispiel

Host Eintrag: 10.100.100.100 / 255.255.255.255

Netzwerkeintrag: 10.100.100.0 / 255.255.255.0

In diesem Beispiel einer Filter-Konfiguration wird der Host-Eintrag ignoriert.

Es gibt zwei verschiedene Möglichkeiten, um einen Eintrag einzufügen:

Betriebssystem	Diese Einträge werden durch den CLI-Befehl FILTER WEBSVR hinzugefügt und sind gültig, bis diese durch den Befehl gelöscht werden. Der Eintrag geht beim Neustart des Systems verloren. Der Befehl kann in der autoexec.lsl nach dem WEBSVR START Kommando eingefügt werden.
Anwendung	Diese Einträge können durch den Aufruf der API von der Anwendung eingefügt werden. Sie werden beim Neustart der Anwendung oder durch einen anderen API-Aufruf ungültig und gelöscht.

2.9.4.2 Authentifizierung mit Benutzername und Passwort

Mit dem LasalOS WEBSVR AUTH CLI-Befehl oder einem API-Aufruf aus der Anwendung können mehrere Webseiten oder ganze Web-Verzeichnisse mit Benutzername- und Kennwort-Authentifizierung geschützt werden.

Der Befehl LasalOS WEBSVR AUTH CLI ist nicht in einer DTC081 CPU verfügbar, da das Betriebssystem den LasalOS Web-Server unterstützen muss.

Benutzername und Passwort sind mit 64-Bit verschlüsselt.

Drei verschiedene Arten der Authentifizierungseinträge sind verfügbar.

Standard	Diese Einträge werden durch das Betriebssystem beim Systemstart ausgeführt und können nicht gelöscht werden. Der Benutzername und das Passwort können jedoch geändert werden. Eine WEBPWD.TXT Datei, welche die Informationen der Einträge in einem codierten Format enthält.
Betriebssystem	Diese Einträge werden bei der Verwendung der WEBSVR AUTH CLI Befehl generiert. Die Betriebssystemeinträge werden ebenfalls in der Datei WEBPWD.TXT gespeichert.
Anwendung	Diese Einträge können durch den Aufruf der API aus der Anwendung generiert werden. Nach dem Aufrufen ist der Eintrag gültig, aber dieser ist nicht in der WEBPWD.TXT Datei gespeichert. Nachdem die Anwendung zurückgesetzt wird, ist der Eintrag ungültig und freigegeben.



Die WEBPWD.TXT Datei wird nur durch LasalOS erstellt und modifiziert. Falls der Inhalt der Datei zerstört oder extern verändert ist, könnte sie ungültige Einträge enthalten, die beim Starten des Web-Servers geladen werden. Wenn der geänderte Eintrag eine LasalOS Einstellung war, gehen alle Betriebssystemeinträge und Änderungen bei Defaultwerten verloren und die Datei wird beim nächsten Start des Web-Servers mit einem Standard-Eintrag erstellt.

2.9.5 Zusätzliche Funktionen

Der LasalOS Web-Server stellt eine zusätzliche DLL (WEBFNC.DLM) zur Verfügung, die mehrere reservierte Callback-Funktionen für die Zukunft enthält.

Das Verzeichnis C:\LSLSYS\HTML ist für Webseiten mit dieser Funktion reserviert.

VERWENDEN SIE NICHT DAS VERZEICHNIS C:\LSLSYS\HTML FÜR IRGENDWELCHE HTML-DATEIEN!

2.9.6 Command Line Interface (CLI) Befehle

2.9.6.1 WEBSVR, WEBSVR INFO

Gibt Informationen über die verfügbaren Befehle des Web-Servers, wenn der Server läuft, sowie über die geladenen DLLs und deren Versionen.

Anforderungen

LasalOS: Ab 01.01.050, für DTC081-IP nicht verfügbar

2.9.6.2 WEBSVR START

Dieser Befehl startet den Web-Server und kann manuell durchgeführt oder von der autoexec.lsl ausgeführt werden. Darüber hinaus erstellt er die WEBPWD.TXT Datei, falls diese nicht vorhanden ist bzw. verändert oder zerstört wurde.

Ist der Befehl fehlgeschlagen, wird eine Fehlermeldung auf dem Bildschirm angezeigt.

Mögliche Gründe für einen Fehler

- Nicht ausreichend Speicher
- Nicht ausreichend freier Speicherplatz zum Erstellen der Datei webpwd.txt
- Die WEBSVR.DLM (C:\LSLSYS) Datei ist nicht verfügbar

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, ab 01.01.103 für DTC081-IP

2.9.6.3 WEBSVR AUTH

Optionale und benötigte Argumente.

INFO

Zeigt Informationen über alle aktuellen Authentifizierungseinstellungen.

Beispiel

```
Web page path/filename Protect.Partition Error 401 File Reference
-----
C:\LSLSYS\HTML      LASALOS      none      Default
C:\WEBROOT          Application   none      Application
C:\WEBROOT\info.htm User          Err401.htm  Operating System
```

Der erste Eintrag ist ein LasalOS-Standardeintrag, der beim Starten des Web-Servers durch das Betriebssystem erstellt wird. Diese Art von Eintrag kann nicht gelöscht werden. Der Benutzername bzw. das Passwort kann jedoch geändert und eine Fehler 401-Datei hinzugefügt werden. Der Eintrag wird in der Datei WEBPWD.TXT gespeichert.

Der zweite Eintrag ist ein Applikationseintrag, der durch den Aufruf einer API-Funktion eingefügt wird. Diese Art von Eintrag wird nicht in der Datei WEBPWD.TXT Datei gespeichert und das LasalOS entfernt den Eintrag beim Reset der Applikation.

Der dritte Eintrag ist ein Betriebssystemeintrag und wird mit dem Befehl WEBSVR AUTH ADD eingefügt. Der Eintrag wird in WEBPWD.TXT-Datei gespeichert und kann mit WEBSVR AUTH REMOVE Befehl entfernt werden, wenn es nicht mehr erforderlich ist.

Alle Einträge in der WEBPWD.TXT Datei werden beim Systemstart geladen.

Wenn z.B. die WEBPWD.TXT Datei durch eine Änderung oder Löschung beschädigt wurde, gehen alle Betriebssystemeinträge verloren. Eine neue WEBPWD.TXT-Datei wird mit den LasalOS Standardeinträgen erstellt.

Die Standardeinstellung für diesen Eintrag ist 5, ohne den LasalOS Standardeintrag. Diese Einstellung kann mit der Variable WEBMAXAUTH erhöht werden.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, nicht für DTC081-IP verfügbar

ADD <filename> <realm> <username:password> <err401_file>

Fügt einen Authentifizierungseintrag ein und speichert diesen codiert in der Datei WEBPWD.TXT.

<filename>	Der Pfad der Web-Seite oder des Dateinamens.
<realm>	Schutzpartition, Name der geschützten Ressource oder Bereich auf dem Server. Die Domain wird in der Dialogbox des Browsers angezeigt, der den Benutzernamen und das Passwort erfordert.
<username:password>	Benutzername und Passwort sind beide erforderlich

Optionale Parameter

<err401_file>	Eine benutzerdefinierte Web-Seite, die bei einem Authentifizierungs-fehler an den Browser gesendet wird.
---------------	----------------------------------------------------------------------------------------------------------

Beispiel

```
WEBSVR AUTH ADD C:\WEBROOT Application Application:appl
```

Alle Dateien in und unter dem Verzeichnis C:\WEBROOT brauchen einen Benutzernamen und ein Passwort. Der Benutzername und das Kennwort müssen jedoch nur einmal eingegeben werden, solange der Browser geöffnet bleibt.

```
WEBSVR AUTH ADD C:\WEBROOT\index.htm Application Application:index
```

In diesem Fall ist ein Benutzername und ein Passwort nur für index.htm Datei erforderlich. Alle anderen Dateien in oder unter dem Verzeichnis C:\WEBROOT brauchen keinen Benutzernamen und Passwort.

```
WEBSVR AUTH ADD C:\WEBROOT\index.htm Application Application:index  
WEBSVR AUTH ADD C:\WEBROOT\ANYDIR Application Application:anydir
```

Für index.htm ist ein Benutzername und ein Passwort erforderlich und alle Dateien in und unter dem Verzeichnis

C:\WEBROOT\ANYDIR brauchen einen Benutzernamen und Passwort.

```
WEBSVR AUTH ADD C:\WEBROOT\index.htm Application Application:index  
WEBSVR AUTH ADD C:\WEBROOT\test.htm Application Application:test
```

Die Dateien C:\WEBROOT\index.htm und C:\WEBROOT\test.htm brauchen einen Benutzernamen und Passwort.

Alle anderen Dateien in oder unter dem Verzeichnis brauchen keinen Benutzernamen und Passwort.

Mit diesem Befehl ist es auch möglich einen Eintrag zu ändern.

Um einen Eintrag zu ändern, müssen die Parameter Dateiname und Domain mit dem bestehenden Eintrag identisch sein. Es können der Benutzername bzw. das Passwort geändert werden oder man fügt eine Fehler 401-Datei ein. Alle Einträge, einschließlich der LasalOS Standardeintrag, können geändert werden.

Ist eine Fehler 401-Datei in dem Eintrag eingegeben, muss diese auf dem Zielrechner oder der Webseite verfügbar sein, wo die benötigte Authentifizierung nicht geladen werden konnte.

Beispiel

```
WEBSVR AUTH ADD C:\WEBROOT Application olduser:oldpass
```

Den oben stehenden Eintrag ändern:

```
WEBSVR AUTH ADD C:\WEBROOT Application olduser:newpass  
WEBSVR AUTH ADD C:\WEBROOT Application newuser:oldpass  
WEBSVR AUTH ADD C:\WEBROOT Application olduser:oldpass err401.htm
```

Die Einträge werden verschlüsselt und Datei WEBPWD.TXT gespeichert.

Die maximale Anzahl der Einträge kann mit Hilfe der Umgebungsvariable WEBMAXAUTH eingestellt werden.

Ohne die LasalOS Standardeinträge, beträgt der Standardwert 5.

Ist der Befehl fehlgeschlagen, wird eine Fehlermeldung auf dem Bildschirm angezeigt.

Mögliche Fehlerursachen:

- Die Liste der Einträge ist voll; erhöhen Sie den Wert der Umgebungsvariable WEBMAXAUTH.

- Der Eintrag ist bereits vorhanden.
- Nicht ausreichend freier Speicherplatz, um der Datei webpwd.txt zu speichern.
- Nicht ausreichend Speicher
- Ungültige oder nicht ausreichende Parameter.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, nicht für DTC081-IP verfügbar

REMOVE <filename> <realm>

Entfernt einen Authentifizierungseintrag und löscht diesen aus der WEBPWD.TXT Datei.

<filename>	Der Pfad der Webseite oder des Dateinamens, der mit dem Befehl ADD eingefügt wird
<realm>	Die mit dem Befehl ADD eingefügte Schutzpartition

Beispiel

```
WEBSVR AUTH REMOVE C:\WEBROOT Application  
WEBSVR AUTH REMOVE C:\WEBROOT\index.htm Application
```

Der Benutzername und das Kennwort sind nicht erforderlich, um einen Eintrag zu entfernen. Ein LasalOS-Standardeintrag kann nicht gelöscht werden.

Ist der Befehl fehlgeschlagen, wird eine Fehlermeldung auf dem Bildschirm angezeigt.

Mögliche Fehlerursachen:

- Der Eintrag existiert nicht
- Nicht ausreichend freier Speicherplatz
- Ungültige oder nicht ausreichende Parameter

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, nicht für DTC081-IP verfügbar

REMOVE ALL

Entfernt alle Authentifizierungseinträge, die vom Betriebssystem und der Applikation zugewiesen sind.

```
WEBSVR AUTH REMOVE ALL
```

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, nicht für DTC081-IP verfügbar

2.9.6.4 WEBSVR FILTER

Optionale und benötigte Argumente.

INFO

Zeigt über alle aktuellen Filter Informationen.

Beispiel

```
WEBSVR FILTER INFO
```

Host/Net	Mask	Referenz
10.100.100.0	255.255.255.0	Operating system
10.10.116.10	255.255.255.255	

Der erste Eintrag ist ein Netzwerkeintrag, der durch das Betriebssystem mit dem Befehl WEBSVR FILTER+ eingefügt wird. Dieser Eintrag ist statisch und kann nur mit dem Befehl FILTER WEBSVR- entfernt werden. Ein Neustart des Systems oder herunterfahren entfernt auch den Eintrag.

Der zweite Eintrag ist ein Anwendungseintrag, der durch einen API-Funktion Aufruf eingefügt wird. Dieser Eintrag wird entfernt, wenn durch die Anwendung ein Reset erfolgt.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, nicht für DTC081-IP verfügbar

+<IPADDR>

Fügt einen Host-Eintrag ein. Der Befehl ist identisch mit dem Befehl unten, aber setzt die Maske automatisch auf 255.255.255.255.

+<IPADDR> <MASK>

Fügt einen Host- oder Netzwerkeintrag ein.

Beispiel

```
WEBSVR FILTER +10.100.100.0 255.255.255.0      ..Netzwerkeintrag  
WEBSVR FILTER +10.10.116.10 255.255.255.255  
WEBSVR FILTER +10.10.116.10                      ..Hosteintrag
```

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, ab 01.01.103 für DTC081-IP

-<IPADDR>

Entfernt einen Host-Eintrag.

Der Befehl ist identisch mit dem Befehl unten, aber setzt die Maske automatisch auf 255.255.255.255.

-<IPADDR> <MASK>

Entfernt einen Host- oder Netzwerkeintrag.

Beispiel

```
WEBSVR FILTER -10.100.100.0 255.255.255.0      ..Netzwerkeintrag  
WEBSVR FILTER -10.10.116.10 255.255.255.255  
WEBSVR FILTER -10.10.116.10                      ..Hosteintrag
```

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, ab 01.01.103 für DTC081-IP

NONE

Entfernt alle Filtereinträge.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, ab 01.01.103 für DTC081-IP

2.9.6.5 WEBSVR STOP

Dieser Befehl stoppt den Web-Server. Der Web-Server kann nur gestoppt werden, wenn es keine aktiven Verbindungen gibt.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL Version ab 01.01.002, ab 01.01.103 für DTC081-IP

2.9.7 Umgebungsvariablen

2.9.7.1 WEBROOT

Das Standardverzeichnis des Web-Servers ist C:\. Der Wert der Variablen muss einen Trailing-Backslash haben.

Um eine Umgebungsvariable zu setzen, muss der Befehl [SETENV\(\)](#) verwendet werden.

Beispiel

SETENV WEBROOT C:\WEBROOT\

Das Standard-Hauptverzeichnis ist C:\.

Die WEBROOT Variable muss gesetzt werden, bevor der Befehl WEBSVR START ausgeführt wird. Nach der Ausführung des Webservers haben Wertveränderungen auf die Variable keine Wirkung.

Das Verzeichnis C:\LSLSYS\HTML wird vom Webserver für die LasalOS-Webseiten verwendet.



2.9.7.2 WEBMAXAUTH

Diese Variable wird zur Erhöhung der Anzahl von Betriebssystem- und Applikations-Authentifizierungseinträgen verwendet. Ohne die LasalOS-Authentisierungseinträge beträgt die Standardeinstellung für diese Variable 5. Die Variable kann auf einen Wert, der weniger als der Standardwert ist, nicht festgelegt werden und muss gesetzt werden, bevor der Webserver aktiv ist.

Änderungen, die während der Ausführung des Webservers gemacht wurden, haben keine Wirkung. Um eine Umgebungsvariable zu setzen ist der Befehl [SETENV\(\)](#) nötig.

Beispiel

```
SETENV WEBMAXAUTH 10
```

2.9.7.3 WEBDEFPWD

Diese Variable wird benutzt, um eine fehlerhafte WebPwd.txt-Datei zu ersetzen. Der Benutzer kann damit entscheiden, ob Authentifizierungsrechte, die in der WebPwd.txt hinterlegt sind, durch Default-Werte ersetzt werden, wenn beim Laden ein Fehler auftritt. Die alte Webpwd.txt wird dann in webpwd.cor umbenannt und eine neue angelegt. Damit die fehlerhafte WebPwd.txt ersetzt wird, muss die Umgebungsvariable auf 1 gesetzt werden. In allen anderen Fällen wird das Laden des WebServers abgebrochen.

Um eine Umgebungsvariable zu setzen, muss der Befehl [SETENV\(\)](#) verwendet werden.

Beispiel

```
SETENV WEBDEFPWD 1
```

2.9.8 Beispielkonfiguration

IPC, C-IPC mit einer Ethernet-Schnittstelle

IP-Adresse	10.10.116.10
Subnetzmaske	255.0.0.0

autoexec.lsl

```
...
SETENV WEBROOT C:\WEBROOT\HTML\
SETENV WEBMAXAUTH 10

WEBSVR START
WEBSVR FILTER +10.10.0.0 255.255.0.0
WEBSVR FILTER +10.100.100.0 255.255.255.0
...
```

C:\WEBROOT\HTML als Hauptverzeichnis für den Server.

Maximale Anzahl von Betriebssystem- und Applikations-Authentifizierungseinträge 10.

Zwei Netzwerkfiltereinträge ermöglichen die Verbindung von 10.10.xx und 10.100.xx

Betriebssystem-Authentifizierungseinträge müssen über Command Line Interface (CLI) eingestellt werden.

```
WEBSVR AUTH ADD C:\WEBROOT\HTML TESTPAGE myname:mypass
```

2.9.9 LASAL OS Web Server API

Zur Verwendung der LasalOS Webserver API, muss ein lsl_webserver Pointer vom Typ LSL_WEBSERVER_API angegeben werden. Die API-Funktion OS_CILGet(), die in lsl_st_ssr.h angegeben ist, muss verwendet werden, um den Zeiger zu initialisieren.

Wird der Zeiger nach dem Aufruf der OS_CILGet()-Funktion gleich NIL, ist die Schnittstelle nicht verfügbar. Dies kann passieren, wenn die OS-Version zu alt ist (<01.01.049), der Webserver nicht läuft oder der Webserver auf der verwendeten Plattform/CPU nicht verfügbar ist. Der Name der Schnittstelle, die für den ersten Parameter der OS_CILGet() benötigt wird ist LSL_WEBSERVER_API.

TYPE

```
    LSL_WEBSERVER_API      : STRUCT
    {
        udVersion          : UDINT;
        udSize              : UDINT;

        pRegisterGetCallback : PVOID;
        pRegisterPostCallback: PVOID;
        pGetLineFromBrowser : PVOID;
        pSendLineToBrowser  : PVOID;
        pFindStringInBuffer : PVOID;
        pSetBrowserList     : PVOID;
        pSetAuthentication  : PVOID;
        pInetStrToByte      : PVOID;
        pGetErrorStringByValue: PVOID;
    };

    END_STRUCT;

```

END_TYPE

Example

```
VAR_GLOBAL
lsl_webserver  :^LSL_WEBSERVER_API;
END_VAR

OS_CILGet( "LSL_WEBSERVER_API" , #lsl_webserver );

if lsl_webserver then
    // ... API call
end_if;
```

Wird eine API-Funktion aufgerufen und der lsl_webserver Zeiger ist gleich NIL, wird die Applikation mit einem ACCESS EXCEPTION Fehler beendet.

2.9.9.1 OS_WEB_FindStringInBuffer

Findet eine Zeichenkette in einem Puffer.

```
FUNCTION GLOBAL __CDECL P_FindStringInBuffer
VAR_INPUT
    pBuffer      : ^CHAR;
    pMatch       : ^CHAR;
    dBufLen      : DINT;
END_VAR
VAR_OUTPUT
    pRetval      : ^CHAR;
END_VAR;

#define OS_WEB_FindStringInBuffer(p1,p2,p3)
    lsl_webserver^.pFindStringInBuffer
    $ P_FindStringInBuffer(p1,p2,p3)
```

Übergabeparameter		Typ	Beschreibung
pBuffer		^CHAR	Zeiger auf den zu suchenden Puffer
pMatch		^CHAR	String, der gesucht werden soll
dBufLen		DINT	Länge des Puffers
Rückgabeparameter		Typ	Beschreibung
pRetval		^CHAR	<p>Wenn erfolgreich, wird ein Zeiger am Beginn der Daten nach dem String pMatch zurückgegeben.</p> <p>Tritt ein Fehler auf, wird ein NIL-Zeiger zurückgegeben.</p>

Die Funktion sucht den Puffer für einen String von "pMatch =..." mit einem der folgenden Zeichen: &, \ 0, \ n

Dadurch wird sichergestellt, dass eine Suche nach Werten zu ApplyValues nicht passt.

Die Funktion wird benutzt, um die Postbefehle von einem Browser mit der API-Funktion [OS_WEB_GetLineFromBrowser\(\)](#) zu verarbeiten.

Beispiel

Eine Suche nach apply in einem Puffer, der &apply=1 enthält, gibt einen Zeiger auf 1 zurück.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL ab Version 01.01.002, Version 01.01.103 für DTC081-IP, Header-Datei : lsl_st_webserver.h

2.9.9.2 OS_WEB_GetErrorStringByValue

Gibt einen Pointer auf eine LasalOS-Fehlermeldung zurück.

```
FUNCTION GLOBAL __CDECL P_GetErrorStringByValue
VAR_INPUT
  dErrValue      : DINT;
END_VAR
VAR_OUTPUT
  pErrString     :^CHAR;
END_VAR;

#define OS_WEB_GetErrorStringByValue(p1)
  lsl_webserver^.pGetErrorStringByValue
  $ P_GetErrorStringByValue(p1)
```

Übergabeparameter	Typ	Beschreibung
dErrValue	DINT	Fehlerwert, der durch eine API-Funktion zurückgegeben wird
Rückgabeparameter	Typ	Beschreibung
pErrString	^CHAR	String zu einer LasalOS-Fehlermeldung

Beispiel

```
VAR
  pErrString     :^CHAR;
  dErrValue      : DINT;
END_VAR

dErrValue := // ... API function call

if dErrValue <> 0 then
  pErrString := OS_WEB_GetErrorStringByValue(dErrValue);
end_if;
```

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL ab Version 01.01.002, Version 01.01.103 für DTC081-IP, Header-Datei : lsl_st_webserver.h

2.9.9.3 OS_WEB_GetLineFromBrowser

Wird verwendet, um die Daten zu lesen, die von einem Browser aus der User Postcallback-Funktion gesendet wurden.

```
FUNCTION GLOBAL __CDECL P_GetLineFromBrowser
VAR_INPUT
  pHandle       : pVoid;
```

```

ppBuffer      : ^pChar;
dWait         : DINT;
dType         : DINT;
END_VAR
VAR_OUTPUT
dRetval       : DINT;
END_VAR;

#define OS_WEB_GetLineFromBrowser(p1,p2,p3,p4)
  lsl_webserver^.pGetLineFromBrowser
  $ P_GetLineFromBrowser(p1,p2,p3,p4)

```

Übergabeparameter	Typ	Beschreibung				
handle0	pVoid	Handle, der an die Callback-Funktion durch den Parameter p_web_io_context übermittelt wird				
ppBuffer	^pChar	Zeiger auf einen Zeiger vom Typ CHAR, um einen Zeiger auf die Daten zu erhalten				
dWait	DINT	Die Zeit in Sekunden die man auf die Daten wartet (Limit: 0-65 S)				
dType	DINT	Art, wie die Daten gelesen werden <table border="1" style="margin-left: 20px;"> <tr> <td>WEBS_G</td> <td>Nächstes verfügbare Paket lesen ET_BUF</td> </tr> <tr> <td>WEBS_G</td> <td>Zu nächstes Zeilenende lesen ET_LINE</td> </tr> </table>	WEBS_G	Nächstes verfügbare Paket lesen ET_BUF	WEBS_G	Zu nächstes Zeilenende lesen ET_LINE
WEBS_G	Nächstes verfügbare Paket lesen ET_BUF					
WEBS_G	Zu nächstes Zeilenende lesen ET_LINE					
Rückgabeparameter	Typ	Beschreibung				
dRetval	DINT	≥0 Anzahl der verfügbaren Byte in den Puffer <0 Negativer Fehler-Code Verwenden Sie die API-Funktion OS_WEB_GetErrorStringByValue() , einen Zeiger auf eine Fehlermeldung zu erhalten. Siehe Kapitel Fehler-Codes für weitere Details.				

Wenn dType WEBS_GET_LINE verwendet wird, liest die Funktion Daten bis "\r\n" gefunden wird oder der Puffer voll ist.

Der Puffer ist 0-terminiert und "\r\n" wird gespeichert. Diese Funktion sollte innerhalb des User Postcallback verwendet werden, um vom Browser geschickte Daten abzurufen.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL ab Version 01.01.002, Version 01.01.103 für DTC081-IP, Header-Datei : lsl_st_webserver.h

2.9.9.4 OS_WEB_InetStrToByte

Wandelt eine String-Adresse in eine Byte-Reihenfolge um.

```
FUNCTION GLOBAL __cdecl P_InetStrToByte
VAR_INPUT
    pStrAddr      : ^CHAR;
    pByteAddr     : ^CHAR;
    udsize        : UDINT;
END_VAR
VAR_OUTPUT
    pRetval       : ^CHAR;
END_VAR;

#define OS_WEB_InetStrToByte(p1,p2,p3)
    lsl_webserver^.pInetStrToByte
    $ P_InetStrToByte(p1,p2,p3)
```

Übergabeparameter	Typ	Beschreibung
pStrAddr	^CHAR	Zeiger auf 0-terminierten String, der die Adresse enthält
pByteAddr	^CHAR	Zeiger auf einen Byte-Array, der die Adresse in Byte-weise enthält
udSize:	UDINT	Größe des Byte-Arrays
Rückgabeparameter	Typ	Beschreibung
pRetval	^CHAR	<p>Wenn erfolgreich, wird ein Zeiger auf das Byte-Array zurückgegeben. Tritt ein Fehler auf wird NIL zurückgegeben.</p> <p>Mögliche Fehlerursachen:</p> <p>Parameter pStrAddr = nil, pByteAddr = nil; udSize <4;</p>

Der von pStrAddr aufgezeigte String muss die Adresse in dotted-Format enthalten.

Beispiel

```
VAR
    ADDR      : ARRAY [0..3] OF CHAR;
    pAddr     : ^CHAR;
END_VAR

pAddr := OS_WEB_InetStrToByte("10.100.100.11", #addr[0], sizeof(addr));
```

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL ab Version 01.01.002, Version 01.01.103 für DTC081-IP, Header-Datei : lsl_st_webserver.h

2.9.9.5 OS_WEB_RegisterGetCallback

Mit einem speziellen Nicht-HTML Code in der HTML-Quellcode Datei ist es möglich, dem Webserver zu erlauben, eine Applikations-Callback-Funktion aufzurufen um einen Browser-get Befehl durchzuführen. OS_WEB_RegisterGetCallback() installiert eine Callback-Funktion.

```
FUNCTION GLOBAL __CDECL P_RegisterGetCallback
VAR_INPUT
    pGetCallback:    : PVOID;
    pThis:          : PVOID;
END_VAR;

#define OS_WEB_RegisterCallback(p1,p2)
    lsl_webserver^.pRegisterGetCallback
    $ P_RegisterGetCallback(p1,p2)
```

Übergabeparameter	Typ	Beschreibung
pGetCallback	PVOID	Zu installierende Callback-Funktion
pThis	PVOID	THIS-Zeiger vom Objekt der Klasse, wo der Callback angegeben ist

Wird die Funktion aufgerufen, wenn ein Parameter = NIL, wird der Callback deinstalliert.

GetCallback

Type of the user get callback function.

```
FUNCTION __CDECL GetCallback
VAR_INPUT
    pFncName:          ^CHAR;
    p_web_io_context: ^LSL_WEB_IO_CONTEXT;
    pParam:            ^CHAR;
END_VAR
VAR_OUTPUT
    dRetVal:          : DINT;
END_VAR
```

Übergabeparameter	Typ	Beschreibung
pFncName	^CHAR	Zeiger auf einen 0-terminierten String, der den Funktionsnamen beinhaltet auf den sich die HTML-Quelldatei bezieht.
p_web_io_context	^LSL_WEB_IO_CONTEXT	Zeiger auf eine LSL_WEB_IO_CONTEXT Struktur, der ein Handle enthält, um Daten mit dem Browser auszutauschen; ein Puffer für die Daten und die Datenlänge.
pParam	^CHAR	Zusätzliche Parameter reservieren
Rückgabeparameter	Typ	Beschreibung

dRetVal	DINT	Rückgabewert des LasalOS
		<p>≥ 0 Das Betriebssystem führt die Funktion nicht neu aus; dies ist nützlich zum einmaligen Senden von Daten, zum Beispiel, beim Öffnen einer Webseite.</p> <p>≥ 1 führt das Betriebssystem die Funktion aus bis dRetVal auf 0 oder <0 gesetzt ist.</p>
		Eine Applikation RESET stoppt den Aufruf der Callback-Funktion.

 Die Callback-Funktion muss mit `__CDECL` definiert werden, ansonsten funktioniert es nicht.

Wenn der Web-Server eine Referenz auf eine Funktion im HTML-Quellcode findet, wird die Callback-Funktion ausgeführt.

Beispiel

...HTML-Quellcode

```
<!---#exec cgi="/ApplTest.fn"--->
```

Diese Zeile muss unter Kommentar sein!

Der Parameter pFncName der GetCallback Funktion zeigt auf den Namen ApplTest.fn.

Für jede gefundene Funktion in einer Webseite wird die gleiche Callback-Funktion ausgeführt.

Die Funktionsnamen, auf denen der Parameter pFncName zeigt, müssen daher zur Ausführung des richtigen Codes für jede Webseite verglichen werden.

 Präfix OS_ wird von der webfnc DLL für LasalOS Rückrufe verwendet.

Es besteht auch die Möglichkeit Daten zur Aktualisierung der Webseite durchgehend zu senden. Die Daten müssen im Puffer von `LSL_WEB_IO_CONTEXT` gespeichert werden.

 Die Größe des von `^LSL_WEB_IO_CONTEXT` aufgezeigtes Puffers, beträgt 1514 Byte.

Wenn mehr Platz benötigt wird, muss ein neuer Pointer zugewiesen werden. Der ursprüngliche Zeiger muss gespeichert und zurückgesetzt werden, bevor die Callback-Funktion zurückgeliefert wird. Der Browser erhält und zeigt alles was von der Funktion geschickt wird. Der Puffer kann auch Skript-Code enthalten, die der Browser dann ausführen wird. Um Daten an den Browser zu senden, muss die API-Funktion [OS_WEB_SendLineToBrowser\(\)](#) verwendet werden.

Die Callback-Funktion darf keine Endlosschleife sein!



Das Kapitel [LasalOS Web-Server API – API-Beispiel](#) zeigt, wie man die Callback-Funktionen verwendet.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL ab Version 01.01.002, Version 01.01.103 für DTC081-IP, Header-Datei : lsl_st_webserver.h

2.9.9.6 OS_WEB_RegisterPostCallback

Das Aktionsattribut der HTML-Objekt Elementform kann verwendet werden, damit der Webserver eine Applikations-Callback-Funktion für einen Browser-Postbefehl aufrufen kann. OS_WEB_RegisterPostCallback() installiert eine Post-Callback-Funktion.

```
FUNCTION GLOBAL __CDECL P_RegisterPostCallback
VAR_INPUT
  pPostCallback : PVOID;
  pThis         : PVOID;
END_VAR;

#define OS_WEB_RegisterPostCallback(p1,p2)
  lsl_webserver^.pRegisterPostCallback
  $ P_RegisterPostCallback(p1,p2)
```

Übergabeparameter	Typ	Beschreibung
pPostCallback	PVOID	Zu installierende Callback-Funktion
pThis	PVOID	THIS-Zeiger vom Objekt der Klasse, wo der Callback angegeben wird

Wird die Funktion aufgerufen, wenn ein Parameter = NIL, wird der Callback deinstalliert.

Postcallback

Type des User-Postcallback.

```
FUNCTION __CDECL PostCallback
VAR_INPUT
  pFncName      : ^CHAR;
  p_web_io_context : ^LSL_WEB_IO_CONTEXT;
  dLen          : DINT;
END_VAR
VAR_OUTPUT
  dRetVal        : DINT;
END_VAR
```

Übergabeparameter	Typ	Beschreibung				
pFncName	^CHAR	Zeiger auf einen Null-terminierten String, der den Funktionsnamen beinhaltet auf den sich die HTML-Quelldatei bezieht.				
p_web_io_context	^LSL_WEB_IO_CONTEXT	Zeiger auf eine LSL_WEB_IO_CONTEXT Struktur, der einen Handle enthält, um Daten mit dem Browser auszutauschen sowie einen Puffer für die Daten und die Datenlänge.				
dLen	DINT	Anzahl der Byte im Postbefehl, die vom Browser geschickt wurden.				
Rückgabeparameter	Typ	Beschreibung				
dRetVal	DINT	Rückgabewert des LasalOS <table border="1" style="margin-left: 20px;"> <tr> <td>≥0</td> <td>Das Betriebssystem führt die Funktion nicht mehr aus.</td> </tr> <tr> <td>≥1</td> <td>führt das Betriebssystem die Funktion aus bis dRetVal auf 0 oder ≤0 gesetzt ist.</td> </tr> </table> <p style="margin-left: 20px;">Diese Funktion sollte nicht durchgehend aufgerufen werden.</p>	≥0	Das Betriebssystem führt die Funktion nicht mehr aus.	≥1	führt das Betriebssystem die Funktion aus bis dRetVal auf 0 oder ≤0 gesetzt ist.
≥0	Das Betriebssystem führt die Funktion nicht mehr aus.					
≥1	führt das Betriebssystem die Funktion aus bis dRetVal auf 0 oder ≤0 gesetzt ist.					



Die Callback-Funktion muss mit `__CDECL` angegeben werden, sonst funktioniert sie nicht.

Wenn der Webserver eine Referenz auf eine Funktion im HTML-Quellcode findet, wird die Callback-Funktion ausgeführt. Beim Vergleich mit der Callback-Funktion, wo die Daten entweder einmalig oder laufend gesendet werden, wird der Postcallback erforderlich, um die Daten vom Browser zu erhalten. Da diese Daten, nachdem sie erhalten wurden, nicht mehr verfügbar sind, ist es nicht vorteilhaft diese Funktion ständig aufzurufen.

Es ist möglich die Seite für einen gewissen Zeitraum innerhalb des Postcallbacks zu aktualisieren. Dies sollte jedoch nur gemacht werden, um die über den Browser

erhaltenen Daten zu quittieren. Um die Daten von einem Browser zu lesen, kann die API-Funktion [OS_WEB_GetLineFromBrowser\(\)](#) verwendet werden. Die API-Funktion [OS_WEB_SendLineToBrowser\(\)](#) kann verwendet werden, um eine Bestätigung zu schicken.

Beispiel

...HTML-Quellcode

```
<form action = "ApplPost.fn" method = "POST" name = "ApplTest">  
</form>
```

In diesem Fall wird der Postcallback, falls installiert, mit dem Parameter pFncName = "ApplPost.fn" aufgerufen.

Für jede Funktion in einer Webseite, wird die gleiche Callback-Funktion ausgeführt.

Die Funktionsnamen, auf denen der Parameter pFncName zeigt, müssen daher zur Ausführung des richtigen Codes für jede Webseite verglichen werden.

Präfix OS_ wird von der webfnc DLL für LasalOS Rückrufe verwendet.



Um die Daten zu quittieren, müssen sie in den Puffer von `LSL_WEB_IO_CONTEXT` gespeichert werden.

Die Größe des Puffers, auf dem `^ LSL_WEB_IO_CONTEXT` zeigt beträgt 1514 Byte.



Wenn mehr Platz benötigt wird, muss ein neuer Zeiger zugewiesen werden.

Der ursprüngliche Pointer muss gespeichert und zurückgesetzt werden, bevor die Callback-Funktion zurückgeliefert wird.

Eine typische Anwendung der Postbefehle ist für Benutzerinformationen (Feedback, Bestellungen, ...).

Die Callback-Funktion darf keine Endlosschleife sein!



Das Kapitel [LasalOS Web-Server API – API-Beispiel](#) zeigt, wie man die Callback-Funktionen verwendet.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL ab Version 01.01.002, Version 01.01.103 für DTC081-IP, Header-Datei : lsl_st_webserver.h

2.9.9.7 OS_WEB_SendLineToBrowser

Wird verwendet, um eine Zeile an einen Browser aus den Benutzer GET oder POST Callback-Funktionen zu senden.

```
FUNCTION GLOBAL __cdecl P_SendLineToBrowser
VAR_INPUT
    pHANDLE      : pVoid;
    dWait        : DINT;
    dType        : DINT;
END_VAR
VAR_OUTPUT
    dRetVal      : DINT;
END_VAR;

#define OS_WEB_SendLineToBrowser(p1,p2,p3)
    lsl_webserver^.pSendLineToBrowser
    $ P_SendLineToBrowser(p1,p2,p3)
```

Übergabeparameter	Typ	Beschreibung				
pHandle	pVoid	Handle, der an die Callback-Funktion durch den Parameter p_web_io_context übermittelt wird.				
dWait	DINT	Wartezeit in Sekunden, damit die früheren gesendeten Daten quittiert werden (Limit: 0-65 s)				
dType	DINT	Type der geschickten Daten <table border="1" style="margin-left: 20px;"> <tr> <td>WEBS_P UT_QUE</td><td>wird verwendet, um die Daten in eine Warteschlange zu stellen</td></tr> <tr> <td>WEBS_P UT_SEN D</td><td>sendet alle Daten in der Warteschlange</td></tr> </table>	WEBS_P UT_QUE	wird verwendet, um die Daten in eine Warteschlange zu stellen	WEBS_P UT_SEN D	sendet alle Daten in der Warteschlange
WEBS_P UT_QUE	wird verwendet, um die Daten in eine Warteschlange zu stellen					
WEBS_P UT_SEN D	sendet alle Daten in der Warteschlange					
Übergabeparameter	Type	Description				
dRetVal	DINT	0 Funktion erfolgreich <0 Negativer Fehler-Code				

Die Funktion kopiert die Byte von Daten, welche in p_web_io_context.^Length_out spezifiziert sind, von p_web_io_context.^Buffer_out in die Output-Warteschlange. Ist kein Platz vorhanden, so werden Daten, die sich in der Warteschlange befinden auf den Browser gesetzt. Wird die GET oder POST Callback-Funktion zurückgegeben, werden alle Daten in der Warteschlange gesendet.

Die Größe des Puffers beträgt 1514 Bytes.



Wird zusätzlicher Platz benötigt, kann der Zeiger im Puffer gespeichert und ein neuer Pointer gesetzt werden. Wenn die Funktion zurückgegeben wird, muss der neue Zeiger durch den alten ersetzt werden.

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL ab Version 01.01.002, Version 01.01.103 für DTC081-IP, Header-Datei : lsl_st_webserver.h

2.9.9.8 OS_WEB_SetAuthentication

Wird zur Änderung oder Einfügung eines Authentifizierungseintrags von der Applikation verwendet.

```
FUNCTION GLOBAL __CDECL P_SetAuthentication
VAR_INPUT
  pFilename      : ^CHAR;
  pRealm         : ^CHAR;
  pUserPwd       : ^CHAR;
  pErr401File   : ^CHAR;
  udFlags        : UDINT;
END_VAR
VAR_OUTPUT
  dRetval        : DINT;
END_VAR;

#define OS_WEB_SetAuthentication(p1,p2,p3,p4,p5)
  lsl_webserver^.pSetAuthentication
  $ P_SetAuthentication(p1,p2,p3,p4,p5)
```

Übergabeparameter	Typ	Beschreibung
pFilename	^CHAR	Pfad der Web-Seite oder des Dateinamens

pRealm	^CHAR	Schutzpartition, Name der geschützten Ressource oder Bereich auf dem Server. Die Domain wird in der Dialogbox des Browsers angezeigt, der den Benutzernamen und das Passwort erfordert.
pUserPwd	^CHAR	Benutzername und Passwort sind beide erforderlich und müssen durch einen Doppelpunkt getrennt werden.
pErr401File	^CHAR	Optional, kann auf NIL gesetzt werden, falls nicht benötigt
udFlags	UDINT	Flags für den Operationstyp. WEBS_A UTH_AD D wird zum Einfügen eines Eintrags verwendet WEBS_B L_REMO VE entfernt einen Eintrag WEBS_B L_REMO VEALL Alle Einträge entfernen WEBS_A UTH_BL OCKING zur Verwendung im Blocking-Modus.
Rückgabeparameter	Typ	Beschreibung
dRetval	DINT	0 Funktion erfolgreich ≠0 Fehler-Code

Ist die Funktion derzeit durch eine andere Aufgabe in Verwendung, ist die WEBS_E_BUSY if WEBS_BL_BLOCKING Funktion nicht gesetzt. Wenn diese gesetzt ist, wird die Funktion so lange blockiert, bis diese von der anderen Aufgabe der Funktion freigegeben worden ist.

Die Flags ADD, REMOVE, REMOVEALL können verwendet oder mit der WEBS_AUTH_BLOCKING Flag kombiniert werden. Ist WEBS_BL_BLOCKING gesetzt, wird die Funktion blockiert, bis diese von der anderen Aufgabe der Funktion freigegeben wird. Die Dauer der Funktion ist von der Menge der Authentisierungseinträge in den Filterlisten abhängig.

ADD, REMOVE und REMOVEALL können nicht kombiniert werden.

Die mit der API-Funktion zugefügten Einträge werden bei einer Applikation RESET entfernt.

Die Einträge werden in der Datei WEBPWD.TXT nicht gespeichert.

Die Einträge werden immer mit pFilename und pRealm verglichen. Um einen Eintrag zu entfernen, pUserPwd und pErr401File können diese NIL gesetzt werden. Um einen Eintrag zu ändern, muss pFilename und pRealm gleich wie der zu ändernde Eintrag sein.

Beispiel

```

// add an entry
rc := OS_WEB_SetAuthentication("C:\WEBROOT\",
  "APPLICATION",
  "user:pass",
  NIL,
  WEBS_AUTH_ADD);

// change an entry
rc := OS_WEB_SetAuthentication("C:\WEBROOT\",
  "APPLICATION",
  "newuser:newpass",
  "err401.htm",
  WEBS_AUTH_ADD);

// remove an entry
rc := OS_WEB_SetAuthentication("C:\WEBROOT\",
  APPLICATION,
  NIL,
  NIL,
  WEBS_AUTH_REMOVE);

```

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL ab Version 01.01.002, Version 01.01.103 für DTC081-IP, Header-Datei : lsl_st_webserver.h

2.9.9.9 OS_WEB_SetBrowserList

Stellt einen Filtereintrag aus der Applikation ein.

```

FUNCTION GLOBAL __CDECL P_SetBrowserList
VAR_INPUT
  pHost      : ^CHAR;
  pNet       : ^CHAR;
  udFlags    : UDINT;
END_VAR
VAR_OUTPUT
  dRetval   : DINT;
END_VAR;

#define OS_WEB_SetBrowserList(p1,p2,p3)
  lsl_webserver^.pSetBrowserList
  $ P_SetBrowserList(p1,p2,p3)

```

Übergabeparameter		
	Typ	Beschreibung
pHost	^CHAR	Zeiger auf ein 0-3 Array vom Typ CHAR für den Host / net
pNet	^CHAR	Zeiger auf ein 0-3 Array vom Typ CHAR für die Maske
udFlags	UDINT	Flags für den Operationstyp
Rückgabeparameter	Typ	Beschreibung
dRetval	DIN	0 Funktion erfolgreich ≠0 Fehler-Code

Wird die Funktion derzeit durch eine andere Aufgabe verwendet, wird WEBS_E_BUSY if WEBS_BL_BLOCKING Funktion nicht gesetzt. Falls diese gesetzt ist, wird die Funktion blockiert, bis diese von der anderen Aufgabe der Funktion freigegeben wird.

Die Flags ADD, REMOVE, REMOVEALL können verwendet oder mit der WEBS_BL_BLOCKING Flag kombiniert werden. Ist WEBS_BL_BLOCKING gesetzt, wird die Funktion blockiert, bis diese von der anderen Aufgabe der Funktion freigegeben worden ist. Die Dauer der Funktion ist von der Menge der Einträge in den Filterlisten abhängig.

ADD, REMOVE und REMOVEALL können nicht kombiniert werden.

Die mit der API-Funktion hinzugefügten Einträge werden bei einem Applikation RESET entfernt.

Die API-Funktion [OS_WEB_InetStrToByte\(\)](#) kann verwendet werden, um eine String-Adresse in eine Byte-Reihenfolge umzuwandeln.

Diese Funktion prüft nicht, ob Einträge bereits existieren.

Beispiel

```

VAR
  rc    : DINT;
  addr : ARRAY [0..3] OF CHAR;
  mask : ARRAY [0..3] OF CHAR;
END_VAR

OS_WEB_InetStrToByte("10.100.100.100", #addr[0], sizeof(addr));
OS_WEB_InetStrToByte("255.255.255.255", #mask[0], sizeof(mask));

rc := OS_WEB_SetBrowserList(#addr[0], #mask[0], WEBS_BL_ADD);

```

Anforderungen

LasalOS: Ab 01.01.050, Web-Server-DLL ab Version 01.01.002, Version 01.01.103 für DTC081-IP, Header-Datei : lsl_st_webserver.h

2.9.10 Quick Review

Die benötigte Zeit der API-Funktionen ist nicht definiert und daher nicht vorhersehbar. Es wird nicht empfohlen, diese Funktion aus dem zyklischen oder Echtzeit-Task abzurufen.

2.9.11 Fehler-Codes

Define	Wert	OS	Bedeutung
WEBS_E_NONE	0	≥ 01.01.049	Kein Fehler
WEBS_E_TASKHANDLE	-1000	≥ 01.01.049	ungültiger Task-Handle, Fehler beim Erstellen des Web-Server Thread
WEBS_E_LISTFULL	-1001	≥ 01.01.049	Liste ist voll (Liste der Filter-, Authentifizierungsliste)
WEBS_E_MEMORY	-1002	≥ 01.01.049	nicht ausreichend Speicher
WEBS_E_NOT_FOUND	-1003	≥ 01.01.049	Nicht gefunden (Filtereintrag, Authentifizierungseintrag)
WEBS_E_POINTER	-1004	≥ 01.01.049	Ungültiger Pointer (NIL)
WEBS_E_EXISTS	-1005	≥ 01.01.049	Bereits vorhanden (Authentifizierungseintrag)
WEBS_E_ACCESS	-1006	≥ 01.01.049	Zugriff verweigert (z.B: der Versuch einen Standard-Authentifizierungseintrag zu entfernen)
WEBS_E_BUSY	-1007	≥ 01.01.049	Derzeit verwendete Funktion (non-blocking-Modus)
WEBS_E_FLAGS	-1008	≥ 01.01.049	Unbekannte Flags (Parameter udFlags)
WEBS_E_FNCDLL	-1009	≥ 01.01.049	Fehler beim Laden / Entladen webfnc.dlm)

2.9.12 API-Beispiel

Ein einfaches Beispiel demonstriert die Verwendung von API-Funktionen und GET bzw. POST Callback-Funktionen.

Beispiel mit HTML und Text Structure-Quellcode.

Drei HTML-Dateien

- index.htm
- ApplCGI.htm
- Err401.htm

2.9.12.1 HTML Quelldatei: index.htm

```
<html>
<head>
<title>Home</title>
</head>

<body>
<p>Home</p>
<hr>
<p><a href="ApplCGI.htm">Application CGI Test</a></p>
</body>

</html>
```

2.9.12.2 HTML Quelldatei: ApplCGI.htm

```
<html>
<head>
<title>Application CGI Test</title>
</head>
<body>
<p>Application CGI Test</p>
<hr>
<p><a href="index.htm">Home</a></p>
<form action="PostSetServer.fn"
method="post"
name="ServerTable"
onSubmit="return SrvValuesTakeOver()"
onReset="return ResetServerValues()"
<div align="left">
<table border="0" width="25%" bordercolor="#000000">
<tr>
<td width="70%" bgcolor="#eeeeee">WebServer objWebServer</td>
<td width="30%" bgcolor="#999999">&nbsp;</td>
</tr>
```

```
<tr>
  <td align="right" colspan = "2" width="50%" bgcolor="#999999">
    ClassSvr - <input text name="ClassSvrCur" size="8">
  </td>
  <td width="50%">
    <input text name="ClassSvr" size="8">
  </td>
</tr>
<tr>
  <td align="right" colspan = "2" width="50%" bgcolor="#999999">
    Svr1 - <input text name="Svr1Cur" size="8"></td>
  <td width="50%">
    <input text name="Svr1" size="8">
  </td>
</tr>
<tr>
  <td align="right" colspan = "2" width="50%" bgcolor="#999999">Svr2 - <input text
    name="Svr2Cur" size="8"></td>
  <td width="50%">
    <input text name="Svr2" size="8">
  </td>
</tr>
<tr>
  <td align="right" colspan = "2" width="50%" bgcolor="#999999">Svr3 - <input text
    name="Svr3Cur" size="8"></td>
  <td width="50%">
    <input text name="Svr3" size="8">
  </td>
</tr>
</table>
</div>
<input type="submit" value="Server setzen">
<input type="reset" value="Rücksetzen">
</form>
<script>
function ResetServerValues()
{
  document.ServerTable.ClassSvr.value = "";
  document.ServerTable.Svr1.value = "";
  document.ServerTable.Svr2.value = "";
  document.ServerTable.Svr3.value = "";
  return false;
}
</script>
<script>
function SrvValuesTakeOver()
{
  var f = document.ServerTable;
  if (f.ClassSvr.value == "")
    f.ClassSvr.value = f.ClassSvrCur.value;
  if (f.Svr1.value == "")
    f.Svr1.value = f.Svr1Cur.value;
  if (f.Svr2.value == "")
    f.Svr2.value = f.Svr2Cur.value;
  if (f.Svr3.value == "")
    f.Svr3.value = f.Svr3Cur.value;
}
```

```
f.Svr3.value = f.Svr3Cur.value;  
}  
</script>  
</body>  
</html>  
<!--#exec cgi="/ApplCGITest.fn"-->
```

2.9.12.3 HTML Quelldatei: Err401.htm

```
<html>  
  
<head>  
<title>Authentication Error 401</title>  
</head>  
  
<body>  
<p>Authentication error 401</p>  
<hr>  
<p>Invalid username and/or password</p>  
</body>  
  
</html>
```

2.9.12.4 LASAL Class Projekt WebServer

```
Klasse:          WebServer  
  
Servers:          ClassSvr      : DINT  (Read/Write)  
                  Svr1         : DINT  (Read/Write)  
                  Svr2         : DINT  (Read/Write)  
                  Svr3         : DINT  (Read/Write)  
  
Methods:          Global        - Init  
                  - Background  
                  Private       - WebServer  (Constructor)  
                  - GetCallback  ( __CDECL )  
                  - PostCallback ( __CDECL )  
                  - ConvertIToA  
                  - COnvertAToI  
  
Variablen:        m_set_filter  : DINT;  
                  m_set_auth    : DINT;  
                  m_perrorstring :^CHAR;  
  
Objektnetzwerk:  ONWebServer  
  
Objekte:          objWebServer - Background Time = 10
```

2.9.12.5 LASAL Class Projekt WebServer Structure Text source code

```
#include "lsl_st_webserver.h"
#include <lsl_st_ifssr.h>
#define NEW_LINE 0x0A // "\n"
VAR_GLOBAL
    lsl_webserver :^LSL_WEBSERVER_API;
END_VAR
FUNCTION WebServer::WebServer //
    VAR_OUTPUT
    ret_code : ConfStates;
    END_VAR
    lsl_webserver := NIL;
    // pointer to the Web server interface
    OS_CILGet("LSL_WEBSERVER_API", #lsl_webserver);
    m_set_filter := 1;
    m_set_auth := 1;
    ret_code:= C_OK;
END_FUNCTION
FUNCTION VIRTUAL GLOBAL WebServer::Init //
    if _firstscan then
        if lsl_webserver then
            // Install get and post callback
            OS_WEB_RegisterGetCallback(#GetCallback(), this);
            OS_WEB_RegisterPostCallback(#PostCallback(), this);
        end_if;
    end_if;
END_FUNCTION
FUNCTION VIRTUAL GLOBAL WebServer::Background //
    VAR_INPUT
        EAX : UDINT;
    END_VAR
    VAR_OUTPUT
        state (EAX) : UDINT;
    END_VAR
    VAR
        rc : DINT;
        addr : ARRAY[0..3] OF CHAR;
        mask : ARRAY[0..3] OF CHAR;
    END_VAR
    // set a filter, convert from string to byte order
    if m_set_filter = 1 then
        if OS_WEB_InetStrToByte( "10.100.100.100", #addr[0], sizeof(addr)) then
            if OS_WEB_InetStrToByte( "255.255.255.255", #mask[0], sizeof(mask)) then
                rc := OS_WEB_SetBrowserList(#addr[0], #mask[0], WEBS_BL_ADD);
                if rc <> WEBS_E_BUSY then
                    if rc < 0 then
                        m_perrorstring := OS_WEB_GetErrorStringByValue(rc);
                    end_if;
                    m_set_filter := 0;
                end_if;
            end_if;
        end_if;
    end_if;
    // add an authentication entry
```

```
if m_set_auth then
  rc := OS_WEB_SetAuthentication( "C:\WEB\HTML\",
                                    "APPLICATION",
                                    "Application:appl",
                                    "C:\WEB\HTML\Err401.htm",
                                    WEBS_AUTH_ADD);

if rc <> WEBS_E_BUSY then
  if rc < 0 then
    m_perrorstring := OS_WEB_GetErrorStringByValue(rc);
  end_if;
  m_set_auth := 0;
end_if;
end_if;
state := READY;
END_FUNCTION
FUNCTION __CDECL WebServer::GetCallback // 
  VAR_INPUT
    pFncName : ^CHAR;
    p_web_io_context : ^void;
    pParam : ^CHAR;
  END_VAR
  VAR_OUTPUT
    dRetval : DINT;
  END_VAR
  VAR
    lsl_web_io_context :^LSL_WEB_IO_CONTEXT;
    pBufferOutStart :^CHAR;
    ServerString : ARRAY[0..10] OF CHAR;
    CharString : ARRAY[0..1] OF CHAR;
    rc : DINT;
  END_VAR;
  dRetval := -1;
  // cast to LSL_WEB_IO_CONTEXT type
  lsl_web_io_context := p_web_io_context$^LSL_WEB_IO_CONTEXT;
  CharString[1] := 0;
  // save start address of the buffer
  pBufferOutStart := lsl_web_io_context^.buffer_out^;
  // reference in the HTML page
  if _strcmp(pFncName, "ApplCGITest.fn") = 0 then
    // java script code to update the form elements of the HTML page
    _strcpy(lsl_web_io_context^.buffer_out^, "<script>");
    CharString[0] := NEW_LINE;
    _strcat(lsl_web_io_context^.buffer_out^, #CharString[0]);
    // ClassSvr
    ConvertItoA(ClassSvr$UDINT, #ServerString[0]);
    // java script understands ' instead of " too
    _strcat(lsl_web_io_context^.buffer_out^,
    "document.ServerTable.ClassSvrCur.value = '");
    _strcat(lsl_web_io_context^.buffer_out^, #ServerString[0]);
    _strcat(lsl_web_io_context^.buffer_out^, "');");
    // Svr1
    ConvertItoA(Svr1$UDINT, #ServerString[0]);
    _strcat(lsl_web_io_context^.buffer_out^,
    "document.ServerTable.Svr1Cur.value = '");
    _strcat(lsl_web_io_context^.buffer_out^, #ServerString[0]);
```

```

._strcat(lsl_web_io_context^.buffer_out^, "';");
// Svr2
ConvertItoa(Svr2$UDINT, #ServerString[0]);
._strcat(lsl_web_io_context^.buffer_out^,
"document.ServerTable.Svr2Cur.value = ''");
._strcat(lsl_web_io_context^.buffer_out^, #ServerString[0]);
._strcat(lsl_web_io_context^.buffer_out^, "';");
// Svr3
ConvertItoa(Svr3$UDINT, #ServerString[0]);
._strcat(lsl_web_io_context^.buffer_out^,
"document.ServerTable.Svr3Cur.value = ''");
._strcat(lsl_web_io_context^.buffer_out^, #ServerString[0]);
._strcat(lsl_web_io_context^.buffer_out^, "';");
// java script code finish
CharString[0] := NEW_LINE;
._strcat(lsl_web_io_context^.buffer_out^, #CharString[0]);
._strcat(lsl_web_io_context^.buffer_out^, "</script>");
(*
above script cope in text format
after ; is no new line
if a new line would be made, a ; is not needed
<script>
document.ServerTable.ClassSvrCur.value = '0';
document.ServerTable.Svr1Cur.value = '0';
document.ServerTable.Svr2Cur.value = '0';
document.ServerTable.Svr3Cur.value = '0';
</script>
*)
// restore the start address of the buffer
lsl_web_io_context^.buffer_out^ := pBufferOutStart;
// length of the data in the buffer
lsl_web_io_context^.length_out^ := _strlen(lsl_web_io_context^.buffer_out^);
// default return value if OS_WEB_SendLineToBrowser returns success
// dRetVal = 500, OS waits 500 ms to the next call
dRetVal := 500;
// send data to the browser
rc := OS_WEB_SendLineToBrowser( lsl_web_io_context^.handle,
                                100,
                                WEBS_PUT_QUE);
if rc < 0 then
// browser probably left the page
    // return value <= 0, OS should not call the function again,
    // until the browser returns to a page where a reference to the
    // function is found
    dRetVal := -1;
end_if;
end_if;
END_FUNCTION
FUNCTION __CDECL WebServer::PostCallback //
VAR_INPUT
    pFnName : ^CHAR;
    p_web_io_context : ^void;
    dlen : DINT;
END_VAR
VAR_OUTPUT

```

```
dRetval : DINT;
END_VAR
VAR
  lsl_web_io_context :^LSL_WEB_IO_CONTEXT;
  InBuffer : ARRAY [0..1513] OF CHAR;
  pInternBuffer :^CHAR;
  i : DINT;
  count : DINT;
  pServerString :^CHAR;
  ServerStr : ARRAY [0..3] OF ARRAY [0..10] OF CHAR;
  index : DINT;
  ServerNames : ARRAY [0..3] OF ARRAY [0..63] OF CHAR;
  j : DINT;
  pBufferOutStart :^CHAR;
END_VAR;
dRetval := 0;
// cast to LSL_WEB_IO_CONTEXT type
lsl_web_io_context := p_web_io_context$^LSL_WEB_IO_CONTEXT;
// form action reference in the HTML page
if _stricmp(pFncName, "PostSetServer.fn") = 0 then
// if the buffer is too small to hold the data, return error
if dLen > sizeof(InBuffer) then
  dRetval := -1;
  return;
end_if;
count := 0;
while 1 do
// read all the data sent by the browser
  i := OS_WEB_GetLineFromBrowser( lsl_web_io_context^.handle,
                                  #pInternBuffer,
                                  100,
                                  WEBS_GET_BUF);
// if OS_WEB_GetLineFromBrowser return error, exit
  if i <= 0 then
    exit;
  end_if;
// if buffer is too small to hold all the data, return
  if (count + i) > sizeof(InBuffer) then
    dRetval := -1;
    return;
end_if;
// copy the data
  _memcpy(#InBuffer[count], pInternBuffer, i$UDINT);
  count := count + i;
  if count >= dLen then
    exit;
  end_if;
end_while;
// data received ?
  if i > 0 then
// null-termination
  InBuffer[count] := 0;
// copy server names to search for
  _strcpy(#ServerNames[0][0], "ClassSvr");
  _strcpy(#ServerNames[1][0], "Svr1");
```

```

    _strcpy(#ServerNames[2][0], "Svr2");
    _strcpy(#ServerNames[3][0], "Svr3");
    j := 0;
    while j < 4 do
    // parse buffer sent by the browser
    pServerString := OS_WEB_FindStringInBuffer( #InBuffer[0],
                                                #ServerNames[j][0],
                                                count);

    index := 0;
    // get value and done by the first separator
    while pServerString^ <> '&' &
        pServerString^ <> 0 &
        pServerString^ <> 0x0a &
        pServerString^ <> 0x0d do
        ServerStr[j][index] := pServerString^;
        index := index + 1;
        pServerString := pServerString + 1;
    end_while;
    ServerStr[j][index] := 0;
    j := j + 1;
    end_while;
// convert server values from ascii to int
    ClassSvr := ConvertAtoi(#ServerStr[0][0]);
    Svr1 := ConvertAtoi(#ServerStr[1][0]);
    Svr2 := ConvertAtoi(#ServerStr[2][0]);
    Svr3 := ConvertAtoi(#ServerStr[3][0]);
// save start address of the buffer
    pBufferOutStart := lsl_web_io_context^.buffer_out^;
// browser changes the site to "PostSetServer.fn" which contains
// data sent by this function
// to remain on current page, just load the page via java script
    // code
    _strcpy(lsl_web_io_context^.buffer_out^, "<script>window.location='ApplCGI.htm';</script>");
    lsl_web_io_context^.length_out^ :=
    _strlen(lsl_web_io_context^.buffer_out^);
// send the buffer to the browser
    OS_WEB_SendLineToBrowser( lsl_web_io_context^.handle,
                            100,
                            WEBS_PUT_QUE);

    end_if;
end_if;
END_FUNCTION
FUNCTION WebServer::ConvertItToA //
    VAR_INPUT
        value0 : UDINT;
        str0 : ^CHAR;
    END_VAR
    VAR
        RevStr : ARRAY[0..10] OF CHAR;
        pRevStr :^CHAR;
        pStr :^CHAR;
        i : USINT;
    END_VAR
    pRevStr := #RevStr[0];
    pStr := str0;

```

```
i := 0;
if value0 <> 0 then
  while value0 do
    pRevStr^ := ('0' + value0 mod 10)$USINT;
    pRevStr += 1;
    value0 := value0 / 10;
    i += 1;
  end_while;
  pRevStr^ := 0;
  pRevStr -= 1;
  while i do
    pStr^ := pRevStr^;
    pRevStr -= 1;
    pStr += 1;
    i -= 1;
  end_while;
  pStr^ := 0;
else
  pStr^ := '0'; pStr += 1;
  pStr^ := 0;
end_if;
END_FUNCTION // WebServer::ConvertItToA
FUNCTION WebServer::ConvertAtoI // 
  VAR_INPUT
    str0 : ^CHAR;
  END_VAR
  VAR_OUTPUT
    value0 : DINT;
  END_VAR
  VAR
    c : DINT;
    total : DINT;
  END_VAR
  value0 := 0;
  if str0 = NIL then
    return;
  end_if;
  total := 0;
  c := str0^; str0 := str0 + 1;
  while c do
    total := 10 * total + (c - '0');
    c := str0^; str0 := str0 + 1;
  end_while;
  value0 := total;
END_FUNCTION
```

2.9.12.6 Einstellungen Autoexec.lsl

```
...
SETENV WEBROOT C:\WEB\HTML\
WEBSVR START
...
```

Die HTML-Quelldateien index.htm, ApplCGI.htm und Err401.htm müssen nach C:\WEB\HTML\ kopiert werden.

2.9.13 Fehler-, Warnung- und Info-Protokollierung

Mit dem Command Line Interface (CLI) Befehl SET DBGLEVEL WEBSVR ist es möglich, einige Info-Protokollierungen einzuschalten.

DBGLEVEL WEBSVR

0	Keine Protokollierung
1	Nur Fehlermeldungen (Standard)
2	Fehlermeldungen und Debug-Meldungen
3	Erweiterte Informationen

Fehlerprotokollierung kann beim Start über die autoexec.lsl oder manuell durch den Command Line Interface (CLI) Befehl aktiviert werden.

Beispiel

SET DBGLEVEL WEBSVR 2
zur Fehlerprotokollierung

2.9.14 Anhang

2.9.14.1 Typen

```
LSL_WEBSERVER_API          : STRUCT
    udVersion             : UDINT;
    udSize                : UDINT;
    pRegisterGetCallback  : PVOID;
    pRegisterPostCallback  : PVOID;
    pGetLineFromBrowser   : PVOID;
    pSendLineToBrowser    : PVOID;
    pFindStringInBuffer   : PVOID;
    pSetBrowserList       : PVOID;
    pSetAuthentication    : PVOID;
    pInetStrToByte        : PVOID;
    pGetErrorStringByValue: PVOID;
END_STRUCT;
LSL_WEB_IO_CONTEXT          : STRUCT
    handle                : PVOID;
    buffer_out             : ^CHAR;
    length_out             : ^UDINT;
END_STRUCT;
```

2.9.14.2 Fehlerwerte

0	WEBS_E_NONE
-1000	WEBS_E_TASKHANDLE
-1001	WEBS_E_LISTFULL
-1002	WEBS_E_MEMORY
-1003	WEBS_E_NOT_FOUND
-1004	WEBS_E_POINTER
-1005	WEBS_E_EXISTS
-1006	WEBS_E_ACCESS
-1007	WEBS_E_BUSY
-1008	WEBS_E_FLAGS
-1009	WEBS_E_FNCDLL

2.9.14.3 Flags

Flags für Filtereinträge

WEBS_BL_ADD	0x00000001	Filtereintrag hinzufügen
WEBS_BL_REMOVE	0x00000002	Filtereintrag entfernen
WEBS_BL_BLOCKING	0x00000004	Blocking-Modus
WEBS_BL_REMOVE_ALL	0x00000008	Alle Einträge entfernen

Flags für die Authentifizierungseinträge

WEBS_AUTH_ADD	0x00000001	Authentifizierungseintrag hinzufügen
WEBS_AUTH_REMOVE	0x00000002	Authentifizierungseintrag entfernen
WEBS_BL_BLOCKING	0x00000004	Blocking-Modus

WEBS_AUTH_Rem OVERALL	0x00000008	Alle Einträge entfernen
--------------------------	------------	-------------------------

Flags zum Senden von Daten an einen Browser

WEBS_PUT_QUE	1	Queuedaten
WEBS_PUT_SEND	2	Alle Daten der Warteschlange senden

Flags zum Lesen von Daten aus einem Browser

WEBS_GET_LINE	1	Ziele bis zur nächsten neuen Zeile abrufen
WEBS_GET_BUF	2	Puffer abrufen

3 OS Interface Library Klassen

3.1 Memory-Library Klassen

3.1.1 RamFile

Die RamFile()-Klasse wird verwendet, um Daten in einer Datei zu speichern. Die Daten werden in der Datei Fdata gespeichert. Wenn die Datei erstellt weniger als 64 kB ist, wird der Inhalt in den RAM geladen, so dass der Lesezugriff wesentlich schneller ist. Der Schreibzugriff wird wie ein normaler Datenzugriff ausgeführt. Ist die Datei größer als 64 kB, werden Daten in die Datei nur geschrieben oder gelesen.

RamFile	
RamFile0	
Setup	m_udLength
2#0000000000000000000000000000000000000000000000000000000000000001101	0
Alarm	FileNameHex
0	16#C06309BB
TaskObjectControl	
0	

3.1.1.1 Schnittstellen

Server

m_udLength	Zeigt die Länge des Datei Daten-Blocks (ohne Header-Daten).
FileNameHex	Diese Zahl dient gleichzeitig als der Datenname und unterscheidet sich bei jedem Objekt.

Clients

Setup	Bit 0:	Muss auf 1 gesetzt, um das RamFile zu starten
	Bit 1:	Muss auf 0 gesetzt sein, um eine Datei zu bearbeiten
	Bit 2:	1: Aktiviert die Checksum-Berechnung
	Bit 3:	1: Die Datei wird codiert
	Bit 4-15	Ist als Initialisierungswert reserviert
Alarm	Auf 1 gesetzt, wenn die Checksum beim Starten nicht der gespeicherten Datei entspricht!	

TaskObjectControl	Automatisch mit dem Betriebssystem Kanal _TaskObjectControl verbunden.
-------------------	------------------------------------------------------------------------

3.1.1.2 Globale Methoden

3.1.1.2.1 GetDataAt

Diese Method wird verwendet, um Daten an beliebiger Stelle aufzurufen. Ein Zeiger auf den Speicherbereich, in dem die Daten gespeichert werden sollen, wird angegeben, nachdem sie gelesen worden sind, sowie die Länge und die Position des ersten Bytes in dem gewünschten Datenblock.

Der Status des Datenzugriffes wird über die Arbeitsverfahren [GetFileState\(\)](#) aufgerufen. Liefert diese Methode den Wert 0 zurück, ist die GetDataAt() Methode abgeschlossen. Für lange Datenblöcke, kann diese Methode viel Zeit benötigen. Daher wird empfohlen, dass die [GetDataAtBackground\(\)](#) Methode verwendet wird.

Übergabeparameter		Typ	Beschreibung
p_us_data			Zeiger auf die gelesenen Daten
du_size			Länge der zu lesenden Daten
du_at			Position, von der aus die Daten gelesen werden sollen
Rückgabeparameter		Typ	Beschreibung
ret_code			Liefert den aktuellen Status zurück

3.1.1.2.2 GetDataAtBackground

Die Methode arbeitet genau wie [GetDataAt\(\)](#), außer dass ein Background-Task zugeordnet ist und die gesamte Datei im Hintergrund verarbeitet wird. Der Status des Datenzugriffes wird mit Hilfe der [GetFileState\(\)](#) Methode aufgerufen. Liefert diese Methode den Wert 0 zurück, ist die GetDataAtBackground() Methode abgeschlossen.

Übergabeparameter		Typ	Beschreibung
p_us_data			Adresse auf der die Daten geschrieben werden
ud_size			Länge der zu lesenden Daten
ud_at			Position, von der aus die Daten gelesen werden sollen
Rückgabeparameter		Typ	Beschreibung
ret_code			Liefert den aktuellen Status zurück

3.1.1.2.3 GetFileState

Der Zugriff auf große Dateien kann oft lange dauern. Um gleichzeitigen Zugriff auf dieselbe Datei zu vermeiden und möglicherweise falsche Daten zu lesen, ist der nächste Datenzugriff nur möglich, nachdem der letzte abgeschlossen ist.

Um festzustellen, ob der letzte Dateizugriff abgeschlossen ist, kann der Status mit Hilfe dieser Funktion abgefragt werden. Liefert diese Methode den Wert 0 zurück, kann die Datei mit der nächsten Methode zugegriffen werden.

Rückgabeparameter	Typ	Beschreibung
State		0 Kein Zugriff auf Dateien 1 Datei wird verarbeitet

3.1.1.2.4 GetUserVersion

Mit Hilfe dieser Methode kann der Benutzer die User-Version von der Header-Datei lesen.

Rückgabeparameter	Typ	Beschreibung
Version	WORT	Anwenderversion

3.1.1.2.5 SetDataAtBackground

Diese Methode arbeitet genau wie die [SetDataAt\(\)](#) Methode, außer dass ein Background-Task zugeordnet wird und der vollständige Datenzugriff im Hintergrund verarbeitet wird.

Der Status des Datenzugriffes wird mit Hilfe der [GetFileState\(\)](#) Methode aufgerufen. Liefert diese Methode den Wert 0 zurück, ist die [SetDataBackground\(\)](#) Methode abgeschlossen.

Übergabeparameter	Typ	Beschreibung
p_us_data		Zeiger auf die zu schreibenden Daten
ud_size		Länge der zu schreibenden Daten
ud_at		Position auf der die Daten geschrieben werden sollen
Rückgabeparameter	Typ	Beschreibung
ret_code		Liefert den aktuellen Status zurück

3.1.1.2.6 SetDataAt

Durch den Aufruf dieser Methode, wird die übertragenen Daten in die Datei geschrieben. Ein Zeiger auf die Daten, die in die Datei geschrieben werden sollten, muss angegeben werden, sowie die Länge und die Position des ersten Bytes in den zu schreibenden Datenblock.

Der Status des Datenzugriffes wird mit Hilfe der [GetFileState\(\)](#) Methode aufgerufen. Liefert diese Methode den Wert 0 zurück, ist die SetDataAt() Methode abgeschlossen.

Mit langen Datenblöcken kann diese Funktion viel Zeit benötigen. Daher wird empfohlen, dass die [GetDataAtBackground\(\)](#) Methode verwendet wird.

Übergabeparameter		Typ	Beschreibung
p_us_data			Zeiger auf die zu schreibenden Daten
ud_size			Länge der zu schreibenden Daten
ud_at			Position auf der die Daten geschrieben werden sollen
Rückgabeparameter		Typ	Beschreibung
ret_code			Liefert den aktuellen Status zurück

3.1.1.2.7 SetSize

Diese Methode wird benutzt, um die Dateigröße zu ändern und sollte nur einmal mit der maximalen Dateigröße aufgerufen werden. Der zyklisch Aufruf dieser Methode reduziert die Lebensdauer der CF deutlich, da die Größe der FAT und des Verzeichnisses bei jeder Änderung aktualisiert werden müssen.

Wenn eine vorhandene Datei mit einem gültigen Namen erweitert wird, wird der hinzugefügte Datenbereich mit einer 0 geschrieben! Wenn die Daten reduziert werden, wird die Datei einfach abgeschnitten und die Daten am Ende der Datei gehen verloren.

Der Status der Datei Zugang ist mit dem [GetFileState\(\)](#) Methode aufgerufen. Wird ein Wert von 0 zurückgegeben, ist die Setsize() Methode abgeschlossen.

Bei Dateien, die größer als 50 KByte sind, kann diese Funktion sehr lange dauern; daher sollte die Funktion [SetSizeBack-ground\(\)](#) verwendet werden.

Übergabeparameter		Typ	Beschreibung
ud_size			Neue Länge der Datei
Rückgabeparameter		Typ	Beschreibung
ret_code			Gibt den aktuellen Status zurück

3.1.1.2.8 SetSizeBackground

Die Methode arbeitet genau wie [SetSize\(\)](#), außer dass ein Background-Task zugeordnet ist und die gesamte Datei im Hintergrund verarbeitet wird. Der Status der Datei Zugang ist mit dem [GetFileState\(\)](#) Methode aufgerufen. Liefert diese Methode einen Wert von 0 zurück, ist die SetSizeBackground() Methode abgeschlossen.

Übergabeparameter	Typ	Beschreibung
ud_size		Neue Länge der Datei
Rückgabeparameter	Typ	Beschreibung
ret_code		Liefert den aktuellen Status zurück

3.1.1.2.9 SetUserVersion

Mit Hilfe dieser Methode kann der Anwender seine eigene User-Version in die Header-Datei schreiben.

Übergabeparameter	Typ	Beschreibung
Version	WORT	Anwenderversion

3.1.1.3 Checksum Berechnung

Um Korruption in einer Datei zu erkennen, kann eine Checksum-Berechnung im Setup Server erstellt werden. Um dies zu ermöglichen, muss Bit 2 gesetzt werden (als Initialisierungswert). Sollte die Checksum-Berechnung deaktiviert werden, muss Bit 2 zurückgesetzt werden (als Initialisierungswert).

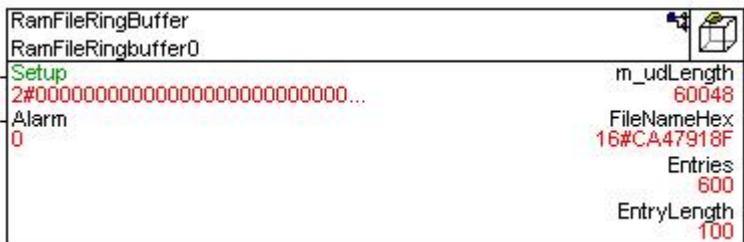
Diese Funktion erhöht den Dateizugriff und verkürzt die Lebensdauer der CF.

3.1.1.4 Encoding

Damit die Datei für jeden Benutzer nicht zugänglich ist, kann sie mit den Setup Server codiert werden. Zur Codierung der Datei, muss Bit 3 gesetzt werden (als Initialisierungswert). Wenn die Datei entschlüsselt werden soll, muss Bit 3 zurückgesetzt werden (als Initialisierungswert).

3.1.2 RamFileRingBuffer

Die RamFileRingBuffer()-Klasse wird zur Speicherung der Datensätze in eine Datei verwendet. Die Daten werden in der Datei Fdata gespeichert. Wenn die erstellte Datei weniger als 64K hat, wird der Inhalt in den RAM geladen, so dass der Lesezugriff wesentlich schneller ist. Der Schreibzugriff wird wie ein normaler Datenzugriff ausgeführt. Ist die Datei größer als 64 kB, werden Daten in die Datei nur geschrieben oder gelesen. Die Anzahl und Länge des Datensatzes werden bei der Erstellung definiert und die aktualisierten Daten am Ende gespeichert. Wenn das Ende der Datei erreicht ist, wird der älteste Wert durch die aktuellen ersetzt, um dafür zu sorgen, dass der aktuellste Datensatz in der Datei gespeichert ist.



3.1.2.1 Schnittstellen

Server

m_udLength	Zeigt die Länge des Datei Daten-Blocks (ohne Header-Daten).
FileNameHex	Diese Zahl dient gleichzeitig als der Datename und unterscheidet sich bei jedem Objekt.
Einträge	Zeigt die maximale Anzahl der Einträge in einer Datei.
EntryLength	Zeigt die Länge eines Eintrags.

Clients

Recordup	Bit 0	Bit 0: Muss auf 1 gesetzt werden, um den RamFile zu starten.
	Bit 1:	Muss auf 0 gesetzt werden, um eine Datei bearbeiten zu können
	Bit 2:	1: aktiviert die Checksum-Berechnung

	Bit 3: 1: Die Datei wird <u>codiert</u>
	Bit 4-15 ist als Initialisierungswert reserviert
Alarm	Wird auf 1 gesetzt, wenn beim Starten die Checksumme mit der gespeicherten Datei nicht übereinstimmt!

3.1.2.2 Globale Methoden

3.1.2.2.1 GetDataRB

Diese Methode wird verwendet, um einen gewünschten Datensatz aufzurufen. Es wird ein Zeiger auf den Speicherbereich angegeben, in dem die aufgerufenen Daten abgelegt werden sollen, sowie die Position des Datensatzes. Der aktuelle Datensatz befindet sich an Position 0, die nächsten älteren Daten an Position 1 usw.

Der Dateizugriffsstatus wird mit der Methode [GetFileState\(\)](#) abgefragt. Liefert diese Methode den Wert 0 zurück, ist die GetDataRB() Methode abgeschlossen.

Mit langen Datenblöcken kann diese Funktion viel Zeit benötigen. Daher wird empfohlen, dass [GetDataRBBackground\(\)](#) verwendet wird.

Übergabeparameter	Typ	Beschreibung
p_us_data		Zeiger auf gelesenen Datensatz
ud_position		Position des Eintrags (0 = aktueller Eintrag)
Rückgabeparameter	Typ	Beschreibung
ret_code		Liefert den aktuellen Status zurück

3.1.2.2.2 GetDataRBBackground

Diese Methode arbeitet wie die [GetDataRB\(\)](#) Methode mit der Ausnahme, dass ein Background-Task zugeordnet wird und der gesamte Dateizugriff im Hintergrund verarbeitet erfolgt. Der Status des Dateizugriffs wird mit der Methode [GetFileState\(\)](#) abgefragt. Liefert diese Methode den Wert 0 zurück, ist die GetDataRBBackground() Funktion abgeschlossen.

Übergabeparameter	Typ	Beschreibung
p_us_data		Zeiger auf die zugegriffenen Daten
ud_position		Position des Eintrags (0 = aktueller Eintrag)

Rückgabeparameter	Typ	Beschreibung
ret_code		Liefert den aktuellen Status zurück

3.1.2.2.3 GetLastDataRB

Diese Methode wird verwendet, um mehrere Datensätze aufzurufen. Ein Zeiger auf den Speicherbereich, in dem die aufgerufenen Daten abgelegt werden sollen, sowie die Anzahl der aufgerufenen Datensätze. Es wird immer der letzte Datensatz gelesen.

Der Dateizugriffsstatus wird mit der Methode [GetFileState\(\)](#) abgefragt. Liefert diese Methode den Wert 0 zurück, ist die [GetDataRB\(\)](#) Methode abgeschlossen. Mit langen Daten Blöcke, kann diese Funktion viel Zeit benötigen. Daher wird es empfohlen, die [GetDataRBBackground\(\)](#) zu verwenden.

Übergabeparameter	Typ	Beschreibung
p_us_data		Zeiger auf die zugegriffenen Daten
ud_anzahl		Anzahl der zu lesenden Einträge (10 = letzten Eintrag)
Rückgabeparameter	Typ	Beschreibung
ret_code		Liefert den aktuellen Status zurück

3.1.2.2.4 GetLastDataRBBackground

Diese Methode arbeitet wie die [GetDataRB\(\)](#) Funktion mit der Ausnahme, dass ein Background-Task zugeordnet wird und der gesamte Dateizugriff im Hintergrund verarbeitet wird. Der Dateizugriffsstatus wird mit der Methode [GetFileState\(\)](#) abgefragt. Liefert diese Methode den Wert 0 zurück, ist die [GetDataRBBackground\(\)](#) Methode abgeschlossen.

Übergabeparameter	Typ	Beschreibung
p_us_data		Zeiger auf die zugegriffenen Daten
ud_anzahl		Anzahl der zu lesenden Einträge (10 = letzten Eintrag)
Rückgabeparameter	Typ	Beschreibung
ret_code		Liefert den aktuellen Status zurück

3.1.2.2.5 GetNumberOfValidEntries

Diese Methode gibt die aktuelle Anzahl von (gültigen) Datensätzen. Wenn der Ring-Puffer wird zum ersten Mal geschrieben, liefert diese Methode der Anzahl der möglichen Einträge (alle Datensätze sind jetzt gültig).

Rückgabeparameter	Typ	Beschreibung
ud_number		Anzahl der gültigen Datensätze

3.1.2.2.6 RecordDataRB

Durch den Aufruf dieser Methode, wird einen neuen Datensatz in die Datei geschrieben. Nur der Pointer auf den Datensatz, der in die Datei geschrieben sollte, wird als Parameter benötigt. Der Dateizugriffsstatus wird mit der Methode [GetFileState\(\)](#) abgefragt. Liefert diese Methode den Wert 0 zurück, ist die RecordDataRB() Methode abgeschlossen. Mit langen Daten Blöcke, kann diese Methode viel Zeit benötigen. Daher wird empfohlen, dass [RecordDataRBBackground\(\)](#) verwendet wird.

Übergabeparameter	Typ	Beschreibung
p_us_data		Zeiger auf den zu schreibenden Datensatz
Rückgabeparameter	Typ	Beschreibung
ret_code		Liefert den aktuellen Status zurück

3.1.2.2.7 RecordDataRBBackground

Diese Methode arbeitet wie die [RecordDataRB\(\)](#) Methode mit der Ausnahme, dass ein Background-Task zugeordnet wird und der gesamte Dateizugriff im Hintergrund verarbeitet wird. Der Dateizugriffsstatus wird mit der Methode [GetFileState\(\)](#) abgefragt. Liefert diese Methode den Wert 0 zurück, ist die RecordDataRBBackground() Methode abgeschlossen.

Übergabeparameter	Typ	Beschreibung
p_us_data		Zeiger auf den zu schreibenden Datensatz
Rückgabeparameter	Typ	Beschreibung
ret_code		Liefert den aktuellen Status zurück

3.1.2.2.8 RecordRingbufferSize

Diese Methode wird benutzt, um die Dateigröße zu ändern und sollte nur einmal mit der maximalen Dateigröße aufgerufen werden. Wird diese Methode zyklisch aufgerufen, wird die Lebensdauer der CF drastisch reduziert, da die Größe vom FAT und Verzeichnis mit jeder Änderung aktualisiert werden müssen. Ist die Datei nicht bereits vorhanden, wird er durch den Aufruf dieser Methode mit den übertragenen Werte erstellt.

Ist die Datei bereits verfügbar und die neue Länge eines Eintrags (ud_length) ist nicht dieselbe wie in der bestehenden Datei, wird regeneriert und es gehen alle alten Daten verloren! Bleibt die Länge eines Eintrags (ud_length) unverändert, werden die Daten einfach wieder hergestellt. Ist die Datei kleiner, wird die Anzahl der Einträge reduziert und die ältesten Einträge gehen verloren. Ist die Anzahl der Einträge erweitert, werden alle Daten beibehalten; die neuen Einträge werden auf 0 gesetzt.

Der Dateizugriffsstatus wird mit der Methode [GetFileState\(\)](#) abgefragt. Liefert diese Methode den Wert 0 zurück, ist die RecordRingbufferSize() Methode abgeschlossen. Bei größeren Dateien (größer als 50 kbytes) kann diese Funktion sehr lange dauern. Daher sollte die Methode [RecordRingbufferSizeBackground\(\)](#) verwendet werden.

Übergabeparameter	Typ	Beschreibung	
ud_entries		Maximale Anzahl der Einträge in der Datei	
ud_length		Länge eines Eintrags	
Rückgabeparameter		Typ	Beschreibung
		ret_code	Liefert den aktuellen Status zurück

3.1.2.2.9 RecordRingbufferSizeBackground

Die Methode arbeitet genau wie [RecordRingbufferSize\(\)](#), außer dass ein Background-Task zugeordnet ist und die gesamte Datei im Hintergrund verarbeitet wird. Der Dateizugriffsstatus wird mit der Methode [GetFileState\(\)](#) abgefragt. Liefert diese Methode den Wert 0 zurück, ist die RecordRingbufferSizeBackground() Funktion abgeschlossen.

Übergabeparameter	Typ	Beschreibung	
du_entries		Maximale Anzahl der Einträge in der Datei	
ud_length		Länge eines Eintrags	
Rückgabeparameter		Typ	Beschreibung
		ret_code	Gibt den aktuellen Status zurück

3.1.2.2.10 RecordUserVersion

Mit Hilfe dieser Methode können die Benutzer ihre eigene Benutzerversion in die Header-Datei schreiben.

Übergabeparameter	Typ	Beschreibung
Version	WORT	Anwenderversion

3.1.2.3 Checksum Berechnung

Um Korruption in einer Datei zu erkennen, kann eine Checksum-Berechnung im Setup Server erstellt werden. Um dies zu ermöglichen, muss Bit 2 gesetzt werden (als Initialisierungswert). Sollte die Checksum-Berechnung deaktiviert werden, muss Bit 2 zurückgesetzt werden (als Initialisierung Wert). Diese Funktion erhöht den Dateizugriff und verkürzt die Lebensdauer der CF.

3.1.2.4 Encoding

Damit die Datei für jeden Benutzer nicht zugänglich ist, kann sie mit den Setup Server codiert werden. Zur Codierung der Datei, muss Bit 3 gesetzt werden (als Initialisierungswert). Wenn die Datei entschlüsselt werden soll, muss Bit 3 zurückgesetzt werden (als Initialisierungswert).

4 LARS

4.1 LARS

4.1.1 Einleitung

Es handelt sich um ein Tool des LASAL-Betriebssystems LasalOS und läuft auf dem Microsoft Windows-System. LARS kann LASAL-Anwendungen mit begrenzter Echtzeit und Hardware-Zugriffsanforderungen ausführen.

4.1.2 Systemanforderungen

LARS kann auf einem IPC oder Standard-PC laufen und benötigt folgende Hard- und Software:

- Windows 2000, ab Microsoft Windows XP
- Pentium 300 MHz Mikroprozessor oder höher
- Minimum 64 MB RAM
- SVGA 800 x 600 Bildschirmauflösung oder höher

Die Hardware-Anforderungen (Prozessor, RAM und Festplattenspeicherplatz) hängen auch von der Speicherkonfiguration von LARS, der Anzahl der laufenden Instanzen auf einem System, der Menge des Speicherplatzes, um LASAL-Projekte zu speichern und der benötigten Ressourcen, die die laufende Applikation unter LARS braucht, ab.

4.1.3 Installation

Führen Sie LarsXX_X.exe aus und folgen Sie den Anweisungen, die auf dem Bildschirm erscheinen. X steht für eine fortlaufende Nummer.

4.1.4 Konfiguration

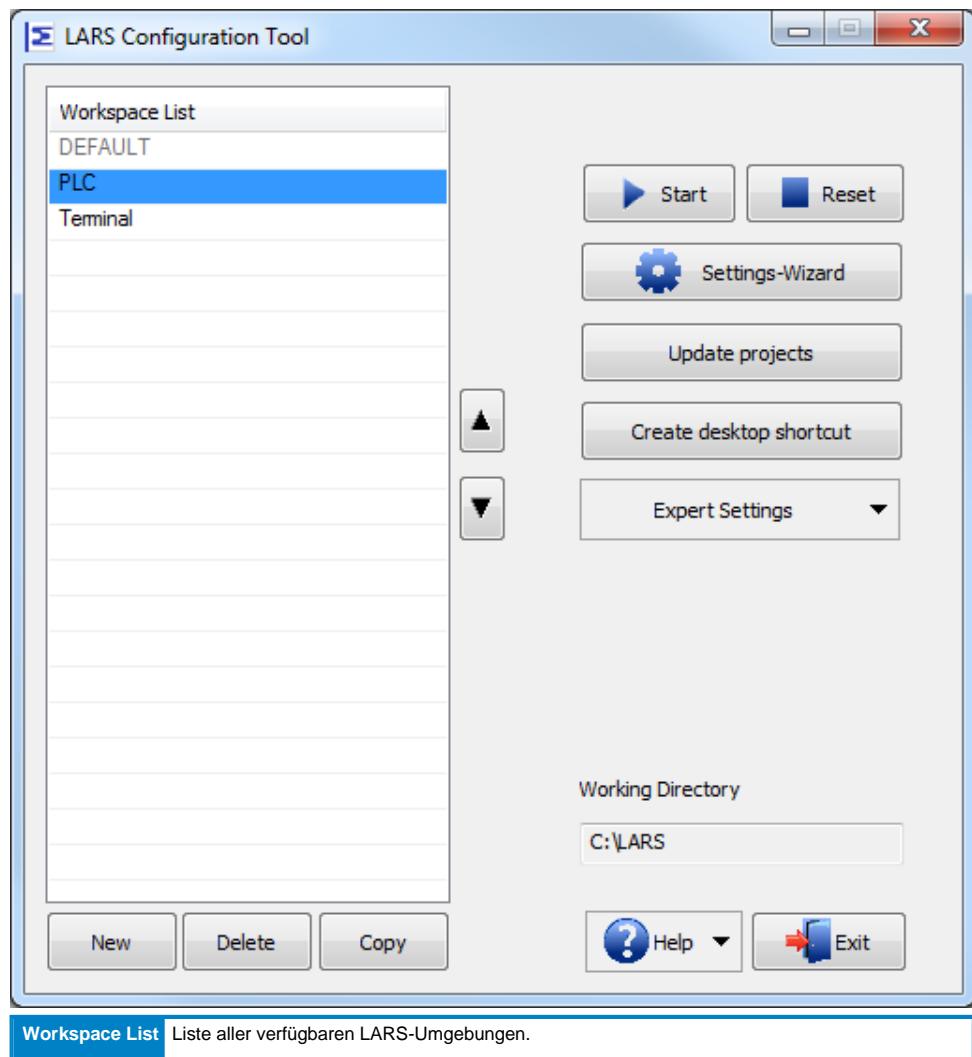
LARS muss mit einem Workspace gestartet werden. Ist kein Workspace angegeben, wird nach einem DEFAULT-Workspace gesucht.

Command Line Parameter

```
/c"<XML-config-file Pfad>"  
/n<Name des LARS-Workspace >  
/s<initial-screenmode> (Bildschirmmodus = WIN oder FULL, Standard = WIN)
```

4.2 LARSConfigTool

Mit dem LARSConfigTool kann man Workspaces für LARS anlegen und parametrieren. Die Informationen werden im %APPDATA%\lasalos2.xml gespeichert.

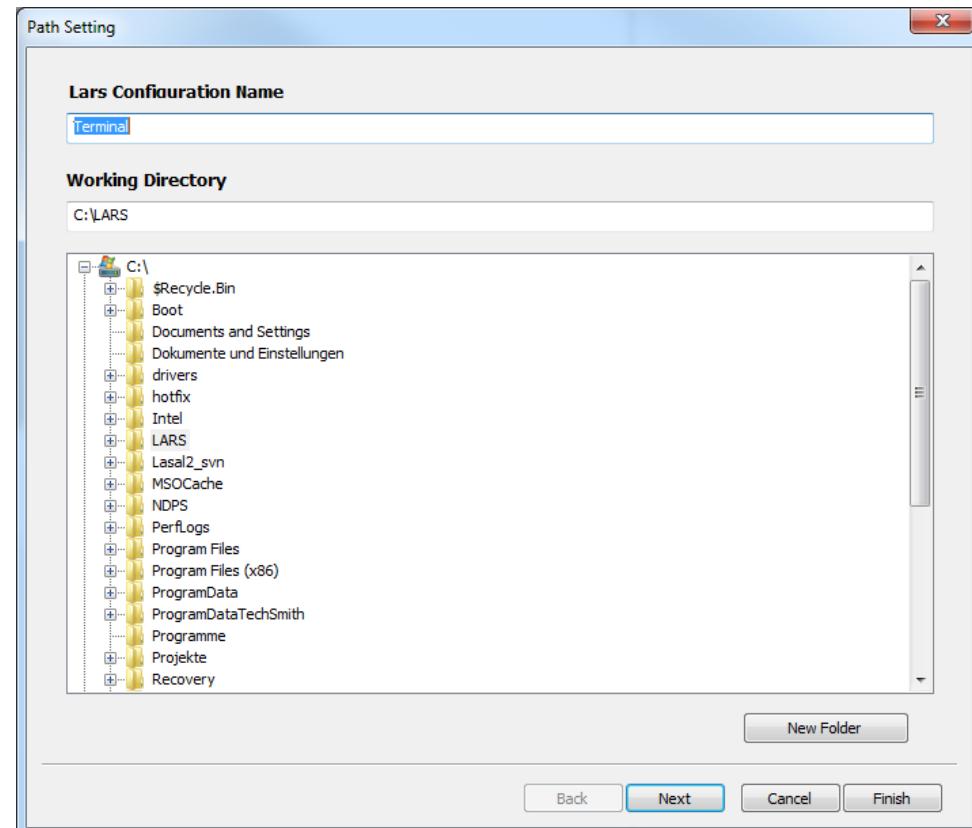


New	Legt einen neuen Workspace an.
Delete	Löscht den markierten Workspace.
Kopieren	Kopiert den markierten Workspace.
Start	Startet das ausgewählte LARS-Projekt.
Reset	Stoppt das laufende LARS-Projekt.
Settings Wizard	Startet die Konfiguration der Einstellungen in Form eines Wizards. Siehe Settings Wizard
Update projects	LARS wird mit dem markierten Workspace gestartet, danach werden die eingestellten Projekte für diesen Workspace aktualisiert. Für diese Funktion müssen LASAL CLASS und LASAL Screen installiert sein.
Create desktop shortcut	Erzeugt einen Shortcut auf dem Desktop vom aktuellen Benutzer für den markierten Workspace.
Expert Settings	Öffnet ein Drop-Down-Menü, in welchem spezielle, selten verwendete Einstellungsmöglichkeiten angeboten werden. Siehe Expert Settings
Help	Ruft wahlweise die englische oder deutsche Hilfe auf bzw. öffnet eine Infobox mit Versionsinformationen.
Exit	Beendet das LARSConfigTool.

4.2.1 **Settings Wizard**

Startet die Konfiguration der Einstellungen in Form eines Wizards.

4.2.1.1 Pfad-Einstellung



Lars Configuration name	Name des Workspace, muss eindeutig sein. Umlaute sind nicht erlaubt, ebenso keine Zahlen am Anfang.
Working Directory	C-Laufwerk für die LARS-Umgebung. Über die Such-Schaltfläche kann ein Pfad ausgewählt werden. Wird dieses C-Laufwerk für die LARS-Umgebung bei einer anderen Konfiguration verwendet, wird darauf durch eine Nachricht hingewiesen:

Warning

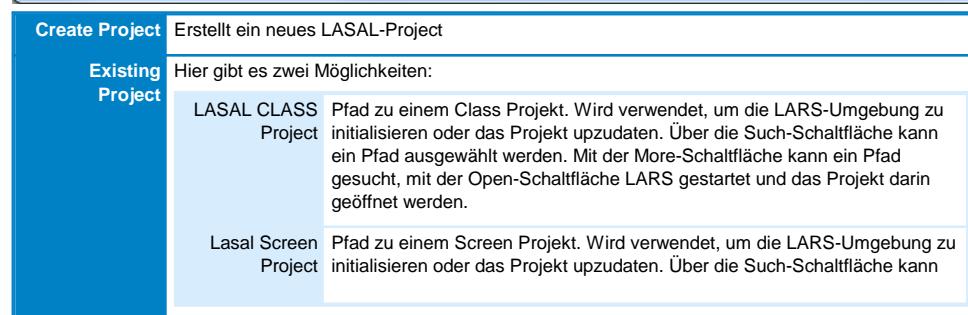
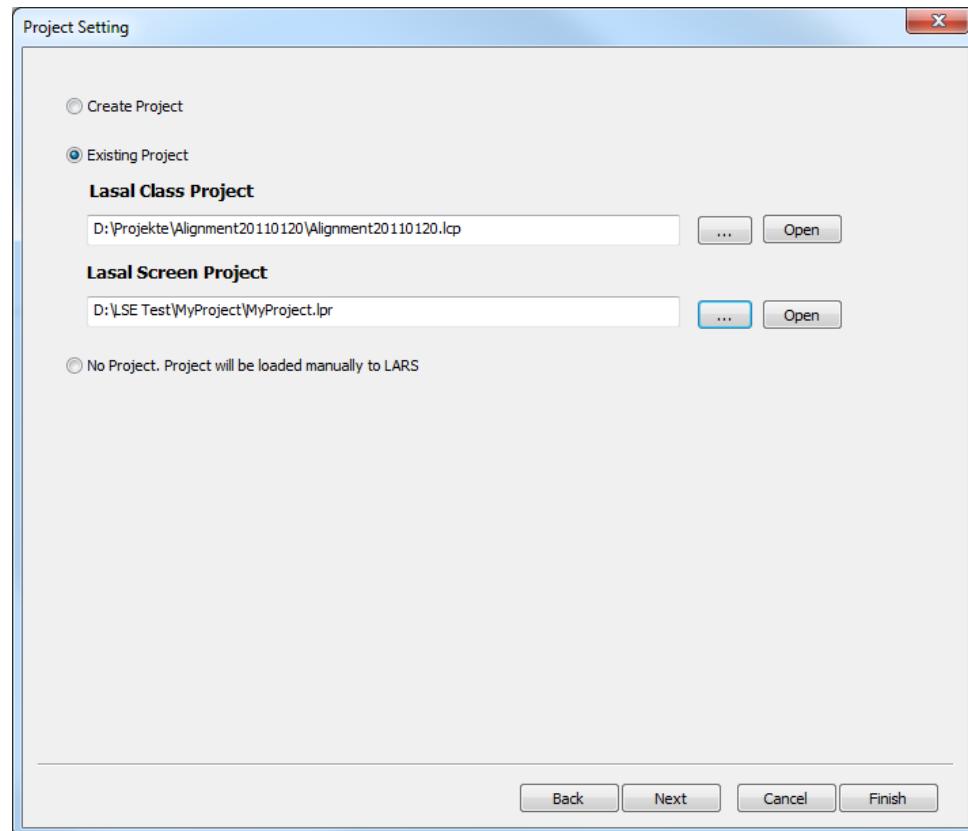


The 'C:\LARS' working directory is already being used by 'DEFAULT' configuration

OK

New Folder	Hiermit kann im aktuell ausgewählten Ordner ein neuer erstellt werden.
Next	Mit der Next Schaltfläche wird das nächste Fenster des Wizards geöffnet:
Cancel	Mit Cancel werden alle Änderungen nach der Bestätigung wieder rückgängig gemacht.
Finish	Mit Finish werden alle Änderungen übernommen (wenn z.B. nur der Name oder der Pfad geändert werden sollte, können die weiteren Fenster hiermit übersprungen werden).

4.2.1.2 Projekt-Einstellung

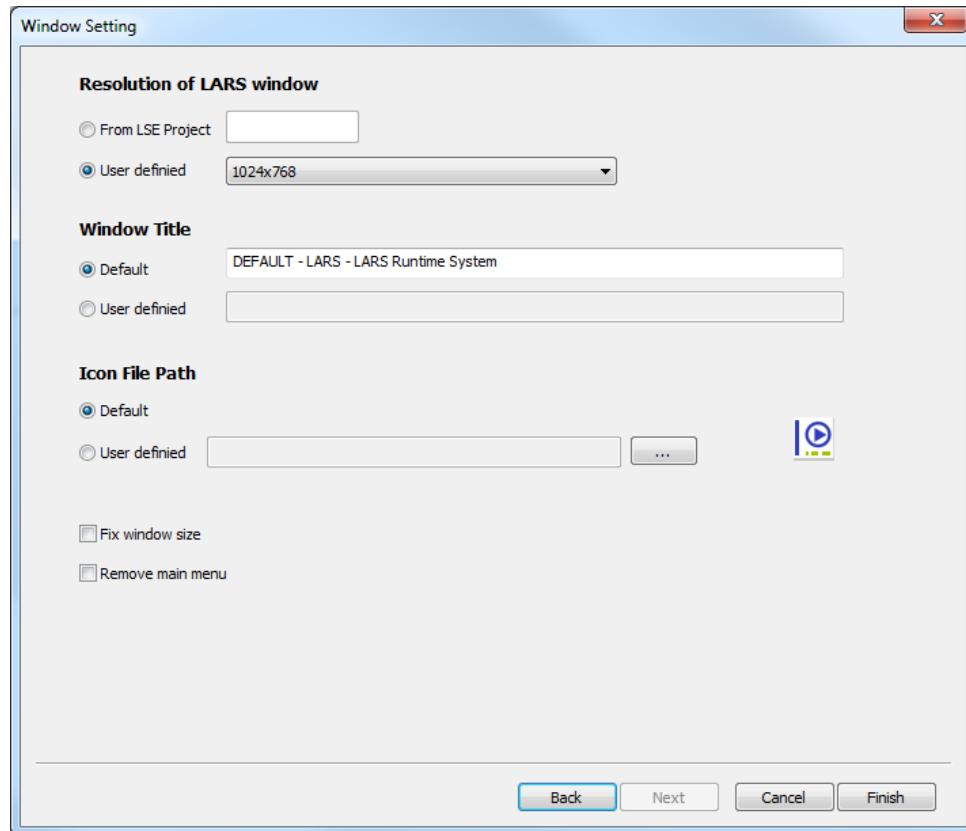


No Project. Das Projekt wird manuell in LARS geladen	<p>ein Pfad ausgewählt werden. Mit der More-Schaltfläche kann ein Pfad gesucht, mit der Open-Schaltfläche LARS gestartet und das Projekt darin geöffnet werden.</p> <p>Der Workspace wird ohne Projekt erstellt. Nach dessen Start muss ein LASAL-Projekt von Hand geladen werden.</p>
-------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Die Buttons am unteren Bildschirmrand haben die gleiche Funktionalität wie im vorherigen Fenster. Mit Next wird das letzte Wizard-Fenster geöffnet:

4.2.1.3 Window-Einstellung

Hier werden die Einstellungen für das Aussehen des Workspace-Fensters getroffen.



Resolution of LARS window	Einstellung für die Auflösung von LARS. Wenn ein Screen-Projekt eingestellt wurde, kann über „From LSE Project“ die Auflösung automatisch ermittelt werden. Mit User defined kann unter vorgegebenen Auflösungen ausgewählt werden.
Window Title	Einstellung für eine alternative Fenster-Bezeichnung. Mit Default wird die Workspace-Bezeichnung als Titel verwendet. Unter User Defined können frei eingebbare Unicode Strings verwendet werden.
Icon File Path	Mit Default wird der voreingestellte Icon Pfad verwendet; unter User Defined kann ein alternativer Icon-Pfad für LARS angegeben oder über die Suchen-Schaltfläche ausgewählt werden.

Fix window size	Wenn aktiviert, ist die Größe des LARS-Fensters nicht mehr veränderbar.
Remove main menu	Wenn aktiviert, wird die LARS-Menüleiste nicht mehr angezeigt.
Finish	Mit der Schaltfläche Finish werden die Änderungen übernommen.

4.2.2 Experteneinstellungen

Öffnet ein Drop-Down-Menü, in welchem spezielle, selten verwendete Einstellungsmöglichkeiten angeboten werden.

4.2.2.1 Reset Window Position

Für jeden Workspace merkt sich LARS die Position beim Beenden. Beim Wechsel von mehreren auf einen Bildschirm kann es passieren, dass LARS sich in einem nicht sichtbaren Bereich öffnet. In diesem Fall muss man mit „Reset workspace position“ die Position vom LARS Workspace wieder auf 0,0 zurückzusetzen.

4.2.2.2 Autoexec.lsl bearbeiten

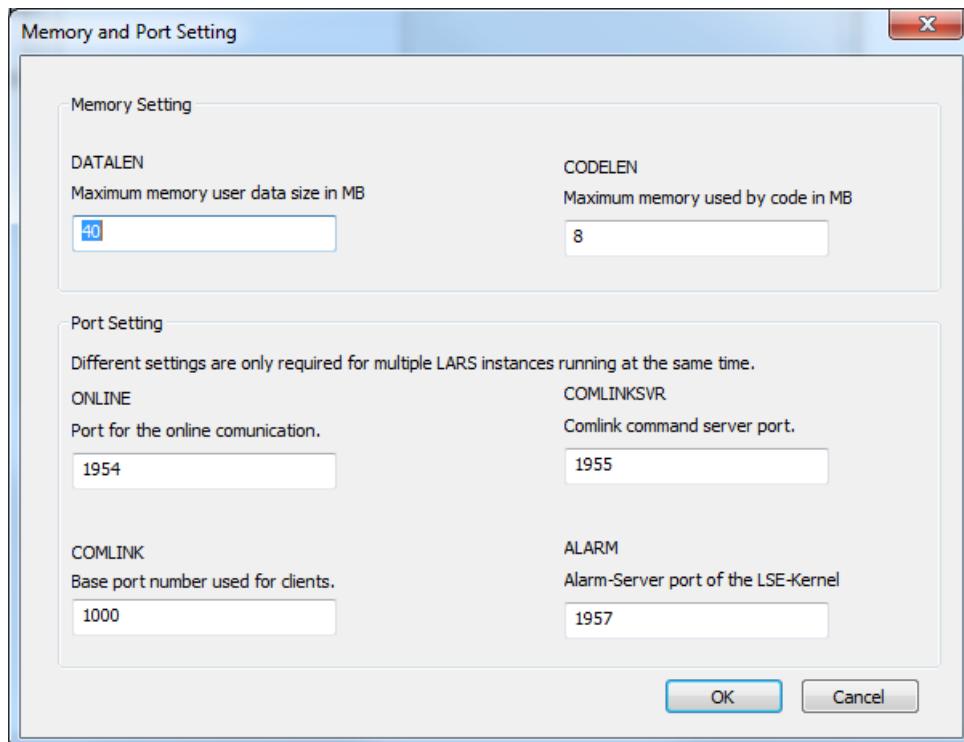
Es wird die Init-Datei Autoexec.lsl in einem Editor zur direkten Bearbeitung geöffnet.

4.2.2.3 Screen-Projekt Server-Stationen pingen

Ist ein Screen-Projekt definiert, in dem entfernte Stationen angegeben wurden, werden diese „angepingt“ und das Resultat in einem Fenster dargestellt:

4.2.2.4 LARS Memory and Port Setting

Es öffnet sich das Memory and Port Setting-Fenster:



DATALEN Gibt den maximalen Benutzer-Datenspeicher an.

CODELEN Gibt den maximalen Codespeicher an.

Die folgende Einstellungen müssen nur dann verändert werden wenn mehrere Lars-Instanzen zur selben Zeit laufen, ansonsten bedarf es keiner Änderung der Einstellungen.

ONLINE TCP/IP-Port-Nummer des Online-Verbindungsservers.

COMLINKSVR TCP/IP-Port-Nummer des Comlink-Servers (Standard: 1955). Der Comlink-TCP/IP-Server benötigt zwei zusätzliche Port-Nummern, ab der in diesem Parameter festgelegten Nummer.

COMLINK TCP/IP-Port-Nummer des internen Programms (Standard: 1000). Diese Port-Nummer dient als Basis für die lokale Port-Nummer verschiedener Systemprogramme (z.B.: TCP/IP-Client in den

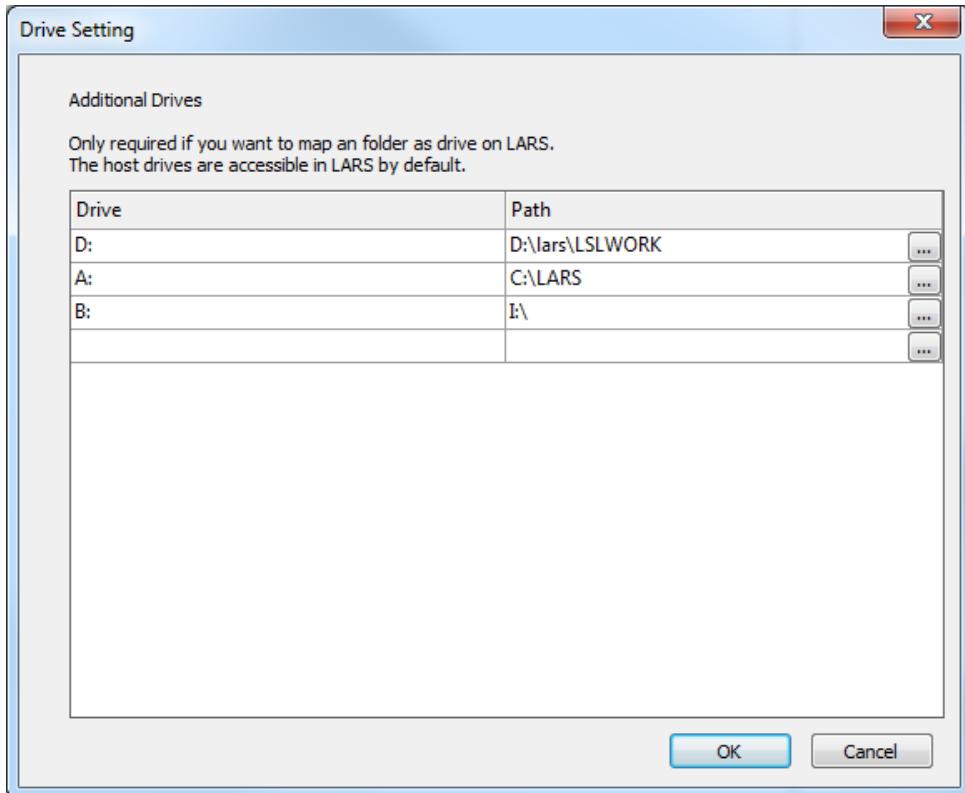
Alarm-Klassen). Diese Systemprogramme fügen ihren eigenen Offset zu dieser Basisnummer hinzu, die nie größer als 10 ist. Das bedeutet, dass diese Basisnummer für eine LARS-Instanz auf +9 reserviert ist. Wird 1000 für die erste LARS-Instanz angegeben, soll die nächste Instanz 1010 als Basis-Port-Nummer haben.

ALARM

TCP/IP-Port-Nummer des Alarm-Servers des LSE-Kernels (Standard: 1957).

4.2.2.5 LARS Drive Mapping

Öffnet den Drive Setting Dialog.

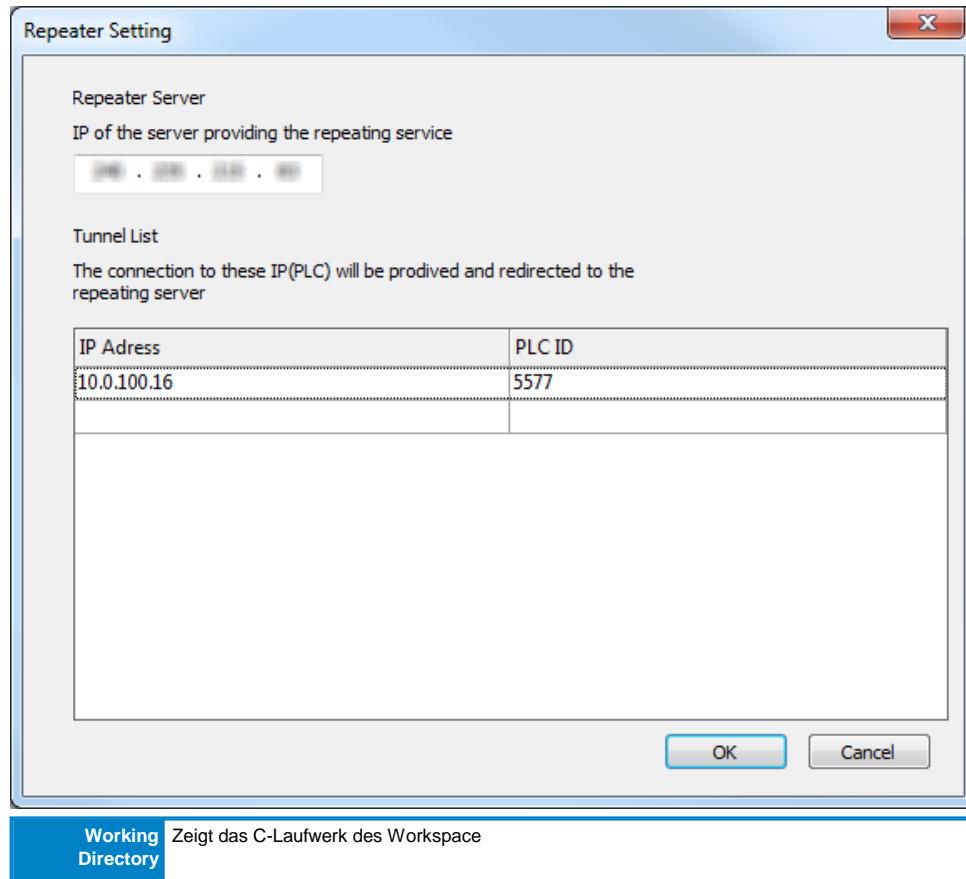


In diesem Dialog können zusätzliche Ordner als Laufwerk in LARS gemappt werden. Die vorhandenen Laufwerke auf dem Host System sind standardmäßig im LARS zugänglich.

In der Drive-Spalte kann der Laufwerksbuchstabe, in der Path-Spalte – mittels Search-Schaltfläche – der Pfad ausgewählt werden.

4.2.2.6 Repeater Connection Configuration

Öffnet den Repeater Setting-Dialog.



4.3 Verwendung von LARS

Nach dem Start von LARS mit der Datei LARS.EXE kann die CLI (Command Line Interface) Befehle ausführen oder eine Online-Verbindung mit LASAL CLASS oder LASAL Screen herstellen. Ein Projekt kann dann heruntergeladen, gespeichert und getestet werden.

Im Screen-Modus kann zwischen einem Fenster mit einem Rahmen und dem Vollbildmodus durch Drücken der Tasten ALT + ENTER oder durch die Wahl von "Ansicht / Vollbild" im Hauptmenü gewechselt werden.

4.3.1 Kommunikationsschnittstellen

Um mit LASAL CLASS oder LASAL Screen online gehen zu können, stehen folgende Schnittstellen zur Verfügung:

- TCP/IP – Das Debug Tool kann entweder auf einem abgesetzten Rechner oder auf dem gleichen Rechner mit LARS ausgeführt werden. Um die von LARS benutzte IP-Adresse zu ermitteln, geben Sie in der CLI den Befehl 'IP' ein.
- RS232
- CAN – LARS kann nur die integrierte CAN-Schnittstelle eines IPCs verwenden. Das CAN-BUS Interface BU 104 kann nicht verwendet werden, da der Controller in dieser Schnittstelle nicht über alle Kommunikationsfunktionen verfügt, die LasalOS benötigt.

Kommunikation zwischen einem Visualisierungs- und einem Maschinenprojekt:

- TCP/IP – Dies ist die bevorzugte Schnittstelle für die Kommunikation zwischen einem Visualisierungs- und einem Maschinenprojekt.
- CAN – LARS kann nur die integrierte CAN-Schnittstelle eines IPCs verwenden.

Kommunikationsschnittstellen, die die APIs (Application Programming Interfaces) verwenden:

- TCP/IP
- CAN – LARS kann nur die integrierte CAN-Schnittstelle eines IPCs verwenden.

4.3.2 Programmierhinweise

Sensitive Anweisungen

Die Intel-Architektur definiert "privilegierte" und "empfindliche" Anweisungen. LARS führt LASAL-Anwendungen auf Privilegebene 3 (CPL = 3) aus, auch User-Modus, Ring 3 oder I/O-Privilegebene 0 (IOPL = 0) genannt. In diesem Punkt unterscheidet sich LARS von LasalOS. LasalOS führt eine LASAL-Anwendung auf IOPL = 3 aus.

Die empfindlichen Anweisungen (auch IOPL-sensitive genannt) können nur ausgeführt werden, wenn $CPL \leq IOPL$ (I/O Privilegebene) ist. Der Versuch, eine empfindliche Anweisung auszuführen, wenn $CPL > IOPL$ ist, erzeugt eine GP-Exception, führt aber nicht zu einem fatalen Fehler. LARS fängt GP-Exceptions, die von der Ausführung empfindlicher Anweisungen verursacht werden, ab und überspringt den betroffenen Befehl.

Die empfindlichen Anweisungen enthalten:

IN	Eingang
INS	Input String
OUT	Ausgang
OUTS	Output String
CLI	Clear interrupt enable flag (IF)
STI	Set IF

Interrupts können daher nicht in LARS deaktiviert werden. Einige Anwendungen versuchen Interrupts zu deaktivieren, um sicherzustellen, dass sich nicht zwei Tasks gleichzeitig im kritischen Bereich befinden. Ein kritischer Bereich ist ein Teil des Programms, der nicht ablaufinvariant ist, z.B. dort, wo auf globale Variablen- oder Hardwareregister zugegriffen wird.

I/O-Anweisungen sind speziell, da sie nicht nur auf IOPL, sondern auch auf das I/O Permission Bitmap empfindlich sind. Das I/O Permission Bitmap wird hinzugezogen, da in LARS das CPL größer ist als das IOPL. Wenn die entsprechenden Bits eines I/O-Ports gelöscht wurden, dann fährt die I/O-Operation fort. Andernfalls wird ein GP-Exception erzeugt, der dann abgefangen und der Befehl übersprungen wird. In der aktuellen LARS-Version (ab Version 5.44) sind alle Bits im I/O Permission Bitmap nicht gleich Null. Dies bedeutet, dass die Ausgabebefehle keine Auswirkung und die Eingabebefehle ein undefiniertes Ergebnis haben.

Diese Einschränkungen müssen beim Schreiben einer neuen Anwendung oder bei der Portierung einer bestehenden Anwendung beachtet werden, um im LARS zu funktionieren.

Intervall von zyklischen Funktionen (CyWork, RtWork oder Background)

Im LARS unterscheidet sich der kleinste Intervall von zyklischen Funktionen vom kleinsten Intervall in einem LASAL-Betriebssystem. In der aktuellen LARS-Version (ab Version 5.44) sind die Werte wie folgt:

- Realtime: 5 ms
- Zyklische, Background: 10 ms

4.3.3 Drucken mit LARS

Es gibt zwei Möglichkeiten, einen Windows-Drucker mit LARS zu verwenden:

1. **Mit dem Menü:**

Diese Option druckt den aktuell sichtbaren Bildschirm auf dem ausgewählten Drucker aus (Pop-up Dialog).

2. **Verwendung der Drucker-Klassen der Applikation:**

Diese Option druckt einen angegebenen Bildschirm (nicht unbedingt der sichtbare Bildschirm) auf dem ausgewählten Drucker aus (Pop-up Dialog). Der Applikationstask, der den Druckauftrag auslöst, wird angehalten, bis der Druckauftrag abgeschlossen ist.

5 Kommunikation und Treiber

5.1 Online-Kommunikation

Die Online-Kommunikation wird zum Datenaustausch zwischen einem auf einem PC gespeichertem Applikationsprogramm (z.B.: Debug Tool) und dem LASAL Betriebssystem benutzt. Eine unmittelbare Kommunikation mit den Objekten eines LASAL-Projekts ist nicht möglich.

Das PC-Programm schickt einen Befehl zum LASAL Betriebssystem und wartet dann auf eine Antwort. Das LASAL Betriebssystem ist nicht in der Lage, Daten zu schicken, es sendet lediglich eine Bestätigung eines Befehls.

Die Online-Kommunikation wird auf dem PC mit der LASAL32.dll realisiert. Diese DLL-Datei exportiert ein Paket von Funktionen, die von einem Windows-Programm aufgerufen werden können. Diese Funktionen werden als LASAL32 API-Funktionen bezeichnet.

Die LASAL32 API-Funktionen sind in folgende Kategorien unterteilt:

- Verwaltung der Online-Verbindung (Verbindung erstellen/entfernen)
- Modem-Funktionen (wählen, auflegen)
- Datenaustausch (Daten empfangen/senden)
- Debug-Funktionen (CPU-Zustand abfragen, Breakpoints, Registerinhalte, Start/Stopp)
- Software-Aktualisierung (Betriebssystem aktualisieren, Module-Download, Progmem)
- Datenübertragung
- Betriebssystemfunktionen (CPU-Belastung, Trace-Aufnahmen, Systemlog)

Implementierung ins Betriebssystem

Für jede Onlineverbindung wird ein eigener Task erstellt. Diese Tasks haben die gleiche Priorität wie zyklische Tasks. Der Kommunikationstask stoppt automatisch, wenn keine Kommunikation über einen bestimmten Zeitraum stattfindet.

5.2 DebugIP-Kommunikation

DebugIP Kommunikation ermöglicht die Kommunikation zwischen einem Applikationsprogramm auf einem PC (z.B. Debugtool) mit einem Command Interpreter im LASAL-Projekt. Am PC ist die DebugIP Kommunikation direkt in der LslOnline.dll und im LASAL CLASS-Code implementiert.

Die folgenden Funktionen stehen für das Applikationsprogramm zur Verfügung:

- Verbindung erstellen/entfernen

- Auf die Adresse eines Objekts mittels Objektnamen zugreifen
- Auf den Klassennamen eines Objekts mittels der Objektadresse zugreifen
- Die Write-Methode eines Servers aufrufen
- Die Read-Methode eines Servers aufrufen

Details

Kommunikationspuffer werden benutzt, um mit dem Command Interpreter zu kommunizieren, wobei ein jedes Debugtool einen eigenen Kommunikationspuffer benutzt. Die einzelnen Felder im Kommunikationspuffer werden von den Funktionen in der LASAL32.dll (SetData, GetData) beschrieben und ausgelesen.

Ein zyklisches Programm läuft im Loader der Steuerung. Es kontrolliert alle aktiven Kommunikationspuffer, ob die Notwendigkeit besteht, ein Kommando auszuführen.

Damit einem Debugtool ein Kommunikationspuffer zugewiesen wird, muss ein I_REGISTER-Kommando über einen Standard-Kommunikationspuffer durchgeführt werden. Dieser Standard-Kommunikationspuffer ist mit einem Verriegelungsmechanismus geschützt, um zu vermeiden, dass mehrere Debugtools gleichzeitig den Puffer benutzen. Wenn ein Kommunikationspuffer für eine bestimmte Zeit nicht aktiv ist, wird er vom Loader entfernt.

Der Kommunikationspuffer steht mit dem Command Interpreter folgendermaßen in Verbindung:

- Das WorkState-Feld des Kommunikationspuffers wird beim Schreiben von WS_QUIT als State definiert.
- Der Befehl wurde übertragen.
- Das WorkState-Feld des Kommunikationspuffers wird auf WS_BUSY gestellt. Sobald der Command Interpreter WS_BUSY in der SPS findet, wird der Befehl ausgeführt.
- Das WorkState-Feld wird so lange in Frage gestellt, bis ein Ergebnis retourniert wird.
- Die Ergebnislänge wird ausgelesen.
- Das Ergebnis wird gelesen.

5.3 Comlink

Daten zwischen Visualisierungen (Clients) und Steuerungen (Server) werden über Comlink ausgetauscht. Der Client schickt entweder einen Befehl zum Server und wartet auf eine Antwort oder fügt eine so genannte UpdateCell in die Updateliste des Servers ein. Eine UpdateCell in der Updateliste befiehlt dem Server, eine Variabel zu überwachen und den Client zu informieren, falls sich der Wert einer Variable ändert.

Der Server enthält zwei Updatelisten - eine statische und eine dynamische. Der Client entscheidet, ob eine UpdateCell in eine statische oder dynamische Updateliste eingefügt werden soll. Normalerweise wird die statische Updateliste beim Projektstart initialisiert und nicht mehr geändert. Die dynamische Updateliste wird aktualisiert, wenn neue Variabellwerte nötig sind (z.B. wenn ein Display mit neuen Variablen ausgewählt wird).

Die Comlink-Schnittstelle enthält folgende Funktionen:

- Login – reserviert einen Kommunikationskanal und erstellt eine Verbindung mit einer lokalen Station oder einer Remote-Station.
- TxCommand – schickt einen Befehl zum Command Interpreter und wartet auf Antwort.
- TxUpd – überträgt eine UpdateCell in eine Updateliste.
- StartStopRefresh – informiert den Server über die Anzahl der UpdateCells in der Updateliste, die gelesen werden soll.
- InstallCallback – installiert eine Callback-Funktion, die aufgerufen wird, wenn sich der Wert einer UpdateCell ändert.

Kommunikationskanal

Daten werden über einem Kommunikationskanal ausgetauscht (Befehle, Antworten und Änderungen in der Updateliste).

Der erste Kanal ist für die lokale Kommunikation vorgesehen. Für die CAN-Kommunikation stehen fünf vordefinierte Kommunikationskanäle zur Verfügung. 16 Kommunikationskanäle sind für TCP/IP vorhanden und werden bei Bedarf zugewiesen.

Applikationsdurchlauf beim Login

Beim Aufruf der Login-Funktion reserviert die Applikation einen Kommunikationskanal und richtet eine Verbindung mit einer lokalen oder einer Remote-Station ein.

CAN

Die Verbindung wird über einen so genannten Channel16 hergestellt. Wobei ein Channel16-Befehl an das Ziel-Betriebssystem über die Stationsobjektnummer 0x700 + Adresse der Zielstation gesendet wird. Der Channel16-Befehl enthält die gewünschte Comlink-Objektnummer der Outstation. Die Comlink-Objektnummer ist die Outstationsnummer +1. Das Betriebssystem der Außenstationen schreibt nach dem Erhalt des Channel16-Befehls die enthaltene Objektnummer in eine Systemvariablen (OPS.CH16buf). Der Loader fragt dann diese Variablen ab. Enthält sie einen gültigen Wert, wird eine Verbindung erstellt.

TCP/IP

Es wird eine Verbindung mit der Outstation-Schnittstelle 1955 (Kommando-Port) und 1956 (Aktualisierungs-Port) erstellt. Die Befehle werden über die Command-Verbindung übertragen und quittiert. Änderungen in der Updateliste werden über die Aktualisierungs-Verbindung empfangen.

RS232

Wird momentan nicht benutzt.

PC: PLC:

6 Online-Kommunikation

6.1 Kommunikationstreiber Lasal32 API

6.1.1 Übersicht

Die Lasal32 Programmierschnittstelle (API) unterstützt die Entwicklung von Anwendungen, die auf einem Betriebssystem zur Kommunikation mit der SPS laufen (läuft auf einem LASAL-Betriebssystem (LasalOS)).

6.1.2 Lasal32 API für Windows

Die Lasal32 API funktioniert nicht auf Betriebssystemen, die älter als Windows 98 sind oder auf Windows NT 4.0 (wie Windows 95, Windows 3.x oder Windows NT 3.xx). Die API ist als eine Dynamic Link Library (DLL) und eine Reihe von Windows-Treibern implementiert.

Die Lasal32 API besteht aus folgenden Dateien:

- Lasal32.dll (Dynamic Link Library)
- Lasal32.lib (Bibliothek mit den exportierten DLL-Funktionen)
- Lasal32.h (C Header-Datei)

6.1.3 Lasal32 API für Linux

Die Lasal32 API besteht aus folgenden Dateien:

- libLasal32.so (Linux Shared Object)
- Lasal32.h (C Header-Datei)

Für Linux wird nur eine TCP-Kommunikation unterstützt.



6.1.4 Lasal32 API-Funktionen

Die Lasal32 API-Funktionen-Dokumentation ist in folgende Kategorien aufgeteilt:

- Verwaltung der Online-Verbindung
- Modem Funktionen

- Datenaustausch
- Status-Informationen und SPS-Befehle
- Administrativfunktionen
- Dateiübertragungen

Ab Lasal32.dll Version 1.34 sind Funktionen thread-sicher. Sie sind mit einem kritischen Abschnitt geschützt. Es ist wichtig zu beachten, dass in früheren Versionen die Anwendung für den Schutz der Funktionen vor wiederholten Aufrufen in Multi-Thread-Anwendungen verantwortlich ist.

6.1.5 Fehler-Behandlung

Typischerweise geben die meisten Lasal32 API-Funktionen einen Fehler durch die Rücklieferung eines Wertes von Null (FALSE) an. Bei einem Fehler rufen die LASAL32 API-Funktionen die Funktion SetLastError auf, um den letzten Fehler-Code für den aktiven Thread einzustellen.

Anwendungen können den von dieser Funktion gespeicherten Wert mittels der GetLastError-Funktion aufrufen, um die Ursache eines Funktionsfehlers zu finden. Der letzte Fehler-Code wird in den lokalen Thread-Speicher gespeichert, um zu verhindern, dass mehrere Threads gegenseitig die Werte überschreiben.

Die folgende Tabelle enthält eine Liste der Fehler-Codes. Sie werden von der GetLastError Funktion zurückgeliefert, wenn viele der Lasal32 API-Funktionen fehlgeschlagen sind.

Code	Beschreibung	Name
0x20000001	Zugriff auf die Schnittstelle wird verweigert.	ERROR_SIGMA32_ACCESSDENIED
0x20000002	Keine Verbindung wurde geöffnet, als Online aufgerufen wurde.	ERROR_SIGMA32_NOTOPEN
0x20000003	Online wurde gerufen, bevor die Verbindung mit dem Aufruf Offline geschlossen wurde.	ERROR_SIGMA32_ALREADYOPEN
0x20000004	Die Funktion unterstützt die ausgewählte Schnittstelle nicht.	ERROR_SIGMA32_NOTDEFINEDFORV24
0x20000005	Der Parameter ist falsch.	ERROR_SIGMA32_WRONGPARAMETER
0x20000006	Ein Schreib Timeout ist aufgetreten.	ERROR_SIGMA32_WRITE_TIMEOUT

0x20000007	Ein Lese Timeout ist aufgetreten.	ERROR_SIGMA32_READ_TIMEOUT
0x20000008	Ein Sendevorgang ist fehlgeschlagen.	ERROR_SIGMA32_WRITE_FAULT
0x20000009	Ein Lesevorgang ist fehlgeschlagen	ERROR_SIGMA32_READ_FAULT
0x2000000C	Eine bestehende Verbindung wurde geschlossen.	ERROR_SIGMA32_CONN_RESET
0x2000000D	Eine Callback-Funktion (CB_FUNCTYPE oder CB_FUNCTYPE2) hat einen Datentransfer mit dem Rückgabewert CBRETURN_ABORT abgebrochen.	ERROR_SIGMA32_ABORTED_BY_USER
0x2000000E	Die Remote-SPS zeigt eine Fehlermeldung an.	ERROR_SIGMA32_REMOTE_ERROR
0x2000000F	Eine Funktion wurde aufgerufen, die nicht von der abgesetzten SPS unterstützt wird.	ERROR_SIGMA32_UNSUPPORTED_CMD
0x20000010	Ein unbekannter allgemeiner Fehler ist aufgetreten.	ERROR_SIGMA32_GENERAL_ERROR
0x20000011	Eine Verbindung konnte nicht hergestellt werden.	ERROR_SIGMA32_CONNECT_FAILED
0x20000012	Unzulässige Funktion.	ERROR_SIGMA32_INVALID_FUNCTION
0x20000013	Timeout wartet auf einen bestimmten CPU-Status.	ERROR_SIGMA32_CPUSTATUS_TIMEOUT
0x20000014L	Kein Speicher mehr vorhanden.	ERROR_SIGMA32_OUT_OF_MEM
0x20000015L	Max. Anzahl der Verbindungen erreicht.	ERROR_SIGMA32_MAX_CONN
0x20000016L	Ungültige ocb Anzahl	ERROR_SIGMA32_INV_OCB_NBR
0x20000017L	Ungültiger Projektzustand	ERROR_SIGMA32_INV_PRJ_STATE
0x20000018L	Puffer ist zu klein	ERROR_SIGMA32_OUT_OF_BUFSIZE
0x20000019L	Puffer ist zu klein	ERROR_SIGMA32_INVALID_DATA
0x2000001AL	Falsche RLB (RefreshListBlock)	ERROR_SIGMA32_INVALID_RLB

0x200000 1BL	Die maximale Länge wurde überschritten	ERROR_SIGMA32_LENGTH_TO_BIG
0x200000 1CL	Falscher VariablenTyp	ERROR_SIGMA32_WRONG_VAR_TYPE
0x200000 1DL	Falsche Länge	ERROR_SIGMA32_LENGTH_WRONG
0x200000 38L	Falsches Online-Passwort	ERROR_SIGMA32_WRONG_ONLINE_PW D
0x200000 40L	Die SPS hat kein Online-Passwort	ERROR_SIGMA32_NO_PASSWORD_SET

6.1.6 Verwaltung der Online-Verbindung

6.1.6.1 IsOnline

Entspricht der [IsOnlineH\(\)](#)-Funktion mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS IsOnline(void);
```

6.1.6.2 IsOnlineH

Die IsOnlineH()-Funktion überprüft eine bestehende Online-Verbindung mit der SPS.

```
extern "C" LSL_BOOL LASAL32_EXPORTS IsOnlineH(int32_t ocbNum);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
Rückgabeparameter	Typ	Beschreibung
		<p>TRUE Erfolg; Verbindung ist aktiv</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die GetLastError()-Funktion auf.</p>

6.1.6.3 LsIPing

Sendet ein Ping-Signal an einen Remote-Host und überprüft die Antwort.

```
extern "C" int32_t LASAL32_EXPORTS LsIPing(
const char* host,
DWORD dwReadTimeout,
SLSIPingInfo* pInfo
);
```

Übergabeparameter	Typ	Beschreibung						
host	const char*	IP-Adresse des Remote-Host als ASCII-String						
dwReadTimeout	DWORD	Timeout-Zeit bis zum Eintreffen einer Antwort in [ms]						
pInfo	SLSIPingInfo*	Zeiger auf eine Variable des Typs SLSIPingInfo. Vor dem Aufruf der Funktion muss im Feld dwSize die Größe der Struktur eingetragen werden.						
Rückgabeparameter	Typ	Beschreibung						
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td><td style="width: 10%;">0</td><td>Erfolg</td></tr> <tr> <td></td><td><0</td><td>Fehler oder Timeout</td></tr> </table>		0	Erfolg		<0	Fehler oder Timeout
	0	Erfolg						
	<0	Fehler oder Timeout						

- 0 LSL_PING_ERROR_NO
- 1 LSL_PING_ERROR_TIMEOUT
- 2 LSL_PING_ERROR_ICMP_CREATE
- 4 LSL_PING_ERROR_INIT
- 5 LSL_PING_ERROR_HOST_NOT_FOUND
- 6 LSL_PING_ERROR_WSA
- 7 LSL_PING_ERROR_GETSTATE
- 8 LSL_PING_ERROR_SHUTDOWN
- 9 LSL_PING_ERROR_RECEIVE_DATA
- 10 LSL_PING_ERROR_RECEIVE_STATUS
- 11 LSL_PING_ERROR_RECEIVE_SIZE
- 12 LSL_PING_ERROR_RECEIVE_PACKETS

Beispiel

```
SLSIPingInfo pingInfo;
```

```
memset(&pingInfo, 0, sizeof(pingInfo));
pingInfo.dwSize = sizeof(pingInfo);
int nRes = LslPing("10.10.170.190", 1000, &pingInfo);
printf("Ping Result=%d Time=%u, TTL=%u\n", nRes, pingInfo.dwNeedTime, pingInfo.dwTTL);
```

6.1.6.4 OcbClose

Gibt die Ressourcen, welche mit [OcbOpen\(\)](#) angelegt/belegt wurden, wieder frei.

```
extern "C" extern "C" void LASAL32_EXPORTS OcbClose(
int32_t ocbNum
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())

6.1.6.5 OcbOpen

Öffnet einen Online Control Block und returniert den Handle dafür. Maximal sind 64 Handles möglich. -1 im Fehlerfall.

Mit diesen OCBs sind mehrere Online-Verbindungen pro DLL-Instanz möglich. Wenn der OCB nicht mehr benötigt wird, muss er wieder freigegeben werden.

```
extern "C" int32_t WINAPI OcbOpen();
```

6.1.6.6 Offline

Entspricht [OfflineH\(\)](#) mit ocbNum = 0

```
extern "C" void LASAL32_EXPORTS Offline(void);
```

6.1.6.7 OfflineH

Die OfflineH()-Funktion deaktiviert eine Online-Verbindung mit der ocbNum übergebenen Verbindung.

Auch wenn Online() fehlschlägt, muss danach Offline() aufgerufen werden.



```
extern "C" void LASAL32_EXPORTS OfflineH(int32_t ocbNum);
```

Übergabeparameter	Typ	Beschreibung

ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
--------	---------	------------------------------------------------------------------------------

6.1.6.8 Online

Die Online()-Funktion stellt eine Verbindung zu einer abgesetzten SPS her.

```
extern "C" LSL_BOOL LASAL32_EXPORTS Online(
    const char *szComm,
    BYTE uBaudRate,
    BYTE uPGStation,
    BYTE uPLCStation,
    BYTE uAutoInit
);
```

Übergabeparameter	Typ	Beschreibung																		
SzComm	const char *	<p>Zeiger auf einen 0-terminierten String, der die Schnittstelle angibt, welche zum Aufbau der Verbindung verwendet wird.</p> <table> <tr> <td>COM1</td><td>X3-X4</td></tr> <tr> <td>COM2</td><td>X3-X4</td></tr> <tr> <td>COM3</td><td>X3-X4</td></tr> <tr> <td>COM4</td><td>X3-X4</td></tr> <tr> <td>COMM</td><td>Vor der Herstellung einer Verbindung über ein Modem muss eine aktive Modemverbindung vorhanden sein. Siehe auch: Funktionen zum Verwalten einer Modemverbindung.</td></tr> <tr> <td>TCP/IP</td><td>TCP:<IP Address>[:<Port>][;ROUTE=<Route-Nr>] <IP-Adresse> definiert die IP-Adresse in der Standard Internet ". Notation (dotted). <Port> definiert die Port-Nummer, die von der abgesetzten SPS verwendet wird. Port ist optional, der Standardwert ist 1954. Soll über eine Route online gegangen werden, kann dies über den optionalen Parameter ROUTE geschehen. <Route-Nr> ist die auf der SPS vergebene Route-Nummer.</td></tr> <tr> <td>LPT1</td><td>CAN-Bus Adapter BU104 auf LPT1</td></tr> <tr> <td>LPT2</td><td>CAN-Bus Adapter BU104 auf LPT2</td></tr> <tr> <td>USBCAN</td><td>CAN-Bus Adapter BU106 auf USB.</td></tr> </table> <p>Hier können zusätzliche Optionen in szComm definiert werden. TCP_NODELAY=x</p>	COM1	X3-X4	COM2	X3-X4	COM3	X3-X4	COM4	X3-X4	COMM	Vor der Herstellung einer Verbindung über ein Modem muss eine aktive Modemverbindung vorhanden sein. Siehe auch: Funktionen zum Verwalten einer Modemverbindung.	TCP/IP	TCP:<IP Address>[:<Port>][;ROUTE=<Route-Nr>] <IP-Adresse> definiert die IP-Adresse in der Standard Internet ". Notation (dotted). <Port> definiert die Port-Nummer, die von der abgesetzten SPS verwendet wird. Port ist optional, der Standardwert ist 1954. Soll über eine Route online gegangen werden, kann dies über den optionalen Parameter ROUTE geschehen. <Route-Nr> ist die auf der SPS vergebene Route-Nummer.	LPT1	CAN-Bus Adapter BU104 auf LPT1	LPT2	CAN-Bus Adapter BU104 auf LPT2	USBCAN	CAN-Bus Adapter BU106 auf USB.
COM1	X3-X4																			
COM2	X3-X4																			
COM3	X3-X4																			
COM4	X3-X4																			
COMM	Vor der Herstellung einer Verbindung über ein Modem muss eine aktive Modemverbindung vorhanden sein. Siehe auch: Funktionen zum Verwalten einer Modemverbindung.																			
TCP/IP	TCP:<IP Address>[:<Port>][;ROUTE=<Route-Nr>] <IP-Adresse> definiert die IP-Adresse in der Standard Internet ". Notation (dotted). <Port> definiert die Port-Nummer, die von der abgesetzten SPS verwendet wird. Port ist optional, der Standardwert ist 1954. Soll über eine Route online gegangen werden, kann dies über den optionalen Parameter ROUTE geschehen. <Route-Nr> ist die auf der SPS vergebene Route-Nummer.																			
LPT1	CAN-Bus Adapter BU104 auf LPT1																			
LPT2	CAN-Bus Adapter BU104 auf LPT2																			
USBCAN	CAN-Bus Adapter BU106 auf USB.																			

		<p>x=0 same as TCP_NODELAY_OFF x=1 same as TCP_NODELAY_ON TCP_NODELAY_ON Schaltet den Nagle-Algorithmus zum Senden Koaleszenz ab TCP_NODELAY_OFF Schaltet den Nagle-Algorithmus zum Senden Koaleszenz ein TCP_TIMEOUT=x Gleich wie TCP_RD_TIMEOUT=x and TCP_WR_TIMEOUT=x TCP_RD_TIMEOUT=x Empfang-Zeitabschaltung in Sekunden TCP_WR_TIMEOUT=x Send-Zeitabschaltung in Sekunden PING_TIMEOUT=x Zeit (rtt – Round trip time) bis zum Eintreffen einer Antwort in Millisekunden PING_AMOUNT=x Anzahl der Pings, die beim Verbindungsaufbau gesendet werden CONNECT_TIMEOUT=x Timeout für TCP Connect in Millisekunden. ONLINE_TIMEOUT=x Maximales Alter der letzten erfolgreichen Kommunikation in Millisekunden, ab dem neu ermittelt wird, ob die Online-Verbindung noch besteht. TCP_NOT_IN_LAN Die SPS befindet sich nicht im LAN. Daher kein Ping beim Verbindungsaufbau und ein langes Timeout für den TCP Connect.</p>
uBaudRate	BYTE	<p>Gibt die Baudrate an, mit der die RS232- oder CAN-BUS-Schnittstelle arbeitet oder ein Lese-Timeout für eine TCP/IP Verbindung.</p> <p>RS232</p> <ul style="list-style-type: none"> • 9600 baud • 19200 baud • 38400 baud • 57600 Baud • 115000 baud <p>CAN BUS</p> <ul style="list-style-type: none"> • 615 kb • 500 kb

		<ul style="list-style-type: none"> • 250 kb • 125 kb • 100 kb • 50 kb • 20 kb <p>Dieser Parameter definiert das Timeout einer Lesefunktion für eine TCP/IP-Verbindung in Sekunden. Ein Wert von 0 bedeutet, dass der Standard Lese-Timeout verwendet werden soll.</p> <p>Für einen Modemanschluss hat dieser Parameter keine Bedeutung und sollte auf 0 gesetzt werden.</p>				
uPGStation	BYTE	Anzahl der CAN-Stationen an den PC (0-31). Für andere Anschlussarten ist dieser Parameter ungültig und sollte auf 0 gesetzt werden.				
uPLCStation	BYTE	Anzahl der CAN-Stationen der SPS (0-31). Für andere Anschlussarten ist dieser Parameter ungültig und sollte auf 0 gesetzt werden.				
Uautolnit	BYTE	Parameter ist veraltet und sollte auf 0 gesetzt werden.				
Rückgabeparameter	Typ	Beschreibung				
		<table border="1"> <tr> <td>TRUE</td><td>Erfolg</td></tr> <tr> <td>FALSE</td><td>Fehler</td></tr> </table> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die GetLastError() Funktion auf.</p>	TRUE	Erfolg	FALSE	Fehler
TRUE	Erfolg					
FALSE	Fehler					

Verwenden Sie [Offline\(\)](#), um die Verbindung zu beenden. Eine Applikation sollte immer einen Aufruf von Offline() für jeden Online()-Aufruf haben, auch wenn Online() fehlschlägt, um Verbindungsressourcen an das System zurückzugeben. [IsOnline\(\)](#) wird verwendet, um zu prüfen, ob eine Verbindung noch besteht, nachdem eine Funktion, die eine Verbindung erfordert, fehlgeschlagen ist. Zur Behebung solcher Fehler benutzen Sie [Offline\(\)](#) und versuchen Sie dann die Verbindung neu aufzubauen.

Wenn eine TCP/IP-Verbindung verwendet wird, beendet die abgesetzte Steuerung automatisch die Verbindung, wenn keine Daten innerhalb von 30 s (neuere Betriebssysteme verwenden einen Timeout Wert von 60 s) übertragen werden. Um eine TCP/IP-Verbindung zu führen, müssen alle Funktionen, die eine Verbindung anfordern (z.B. [GetCpuStatus\(\)](#)) in regelmäßigen Abständen aufgerufen werden.

Beispiel

RS232

```
if (Online("COM1", // Online string
           3,      // Baud rate:      56k
           0,      // PG station no.: n/a
           0,      // PLC station no.: n/a
           0      // CAN Auto init: n/a
           ) == TRUE)
    printf( "Online\n" );
else
    printf( "Offline\n" );
```

Modem

```
if (Online("COMM", // Online string
           0,      // Baud rate:      n/a
           0,      // PG station no.: n/a
           0,      // PLC station no.: n/a
           0      // CAN auto init: n/a
           ) == TRUE)
    printf( "Online\n" );
else
    printf( "Offline\n" );
```

TCP/IP

```
if (Online("TCP:10.24.3.199", // Online string
           0,      // Read timeout: Standard
           0,      // PG station No.: n/a
           0,      // PLC station No.: n/a
           0      // CAN auto init: n/a
           ) == TRUE)
    printf( "Online\n" );
else
    printf( "Offline\n" );
```

6.1.6.9 OnlineH

Entspricht der oben beschriebenen [Online\(\)](#)-Funktion mit dem Unterschied, dass man hier noch einen zusätzlichen Parameter `ocbNum` (Online Control Block) angeben kann. `OcbNum` gibt an, welcher der 64 möglichen Online Control Blocks für den Aufbau der Verbindung verwendet werden soll, diese Nummer erhält man mit der [OcbOpen\(\)](#)-Funktion.

```
extern "C" LSL_BOOL LASAL32_EXPORTS OnlineH(
    int32_t      ocbNum,
    const char  *szComm,
    BYTE         uBaudRate,
```

```

    BYTE      uPGStation,
    BYTE      uPLCStation,
    BYTE      uAutoInit
);

```

6.1.6.10 OnlineOptions

Entspricht [OnlineOptionsH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS OnlineOptions(
void      *pData,
uint32_t  dOption
);

```

6.1.6.11 OnlineOptionsH

Optionale Funktionalitäten beim Aufbau der Verbindung setzen.

```

extern "C" LSL_BOOL LASAL32_EXPORTS OnlineOptions(
int32_t   ocbNum,
void      *pData,
uint32_t  dOption
);

```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
pData	*pData	Zeiger auf die Daten				
dOption	uint32_t	Option, die gesetzt werden soll. Zulässig sind: LSL_ONLINE_TCP_CANCEL_CONNECT_FLAG: setzt eine Abbruchbedingung für das TCP_Connect Kommando. Der Zeiger pData wird in den TCP-Control Block eingetragen. pData muss auf eine Boolesche Variable zeigen. Während des TCP-Verbindungsaufbaus wird diese Boolesche Variable zyklisch abgefragt. Sobald sie TRUE enthält, wird der Verbindungsaufbau abgebrochen.				
Rückgabeparameter	Typ	Beschreibung				
		<table border="1"> <tr> <td>TRUE</td><td>Erfolg</td></tr> <tr> <td>FALSE</td><td>Fehler</td></tr> </table> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die GetLastError()-Funktion auf.</p>	TRUE	Erfolg	FALSE	Fehler
TRUE	Erfolg					
FALSE	Fehler					

Beispiel

```
BOOL rc = FALSE;
BOOL CancelTCPConnect = FALSE;
rc = OnlineOptionsH(ocbNum, &CancelTCPConnect, LSL_ONLINE_TCP_CANCEL_CONNECT_FLAG);
```

6.1.6.12 OnlinePwd

Entspricht der [OnlineH\(\)](#)-Funktion mit dem zusätzlichen Parameter onlPwd. Wenn onlPwd = NULL, dann ist das Verhalten ident. Die Passwortprüfung wird nur bei einer TCP/IP-Verbindung durchgeführt. Das Passwort wird im Klartext an die Funktion übergeben.

```
extern "C" LSL_BOOL LASAL32_EXPORTS OnlinePwd(
    const char *szComm,
    BYTE uBaudRate,
    BYTE uPGStation,
    BYTE uPLCStation,
    BYTE uAutoInit,
    const char *onlPwd
);
```

Falls das Passwort falsch war, wird FALSE retourniert und [GetLastError\(\)](#) gibt den Fehler-Code ERROR_SIGMA32_WRONG_ONLINE_PWD zurück.

Wenn Sie versuchen, mit einem Passwort online zu gehen, die SPS aber nicht passwortgeschützt ist, wird der Befehl OnlinePwd() erfolgreich sein, aber es wird keine Verbindung hergestellt und [GetLastError\(\)](#) gibt den Fehler-Code ERROR_SIGMA32_NO_PASSWORD_SET zurück. Mit [IsOnline\(\)](#) kann geprüft werden, ob eine Online-Verbindung besteht.

Beispiel

```
char onlPwd[32] = { 0 };
const char* pPwd = onlPwd;

strcpy(onlPwd, "OnlinePassword");

if (OnlinePwd("TCP:10.10.170.190", 0, 0, 0, 1, pPwd))
{
    if (IsOnline())
    {
        if (LslReadDateTime(&SysTime))
        {
            printf("%d.%d.%d\n", SysTime.wDay, SysTime.wMonth, SysTime.wYear);
            printf("%d:%d:%d\n", SysTime.wHour, SysTime.wMinute, SysTime.wSecond);
        }
    }
    else
        printf("LslReadDateTime() failed(0x%08X)\n", GetLastError());
    printf("\n");
```

```
    }
    else
        printf("OnlinePwd failed(0x%08X)\n", GetLastError());
}
else
    printf("OnlinePwd failed(0x%08X)\n", GetLastError());
Offline();
```

6.1.6.13 OnlinePwdH

Entspricht der [OnlineH\(\)](#)-Funktion mit der oben beschriebenen Passwort-Funktionalität erweitert.

```
extern "C" LSL_BOOL LASAL32_EXPORTS OnlinePwdH(
    int32_t      ocbNum,
    const char*  szComm,
    BYTE         uBaudRate,
    BYTE         uPGStation,
    BYTE         uPLCStation,
    BYTE         uAutoInit,
    const char*  onlPwd
);
```

6.1.6.14 OnlineSSL

Entspricht OnlineSSLH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS OnlineSSL(
    const char*  szComm,
    uint8_t      uBaudRate,
    uint8_t      uPcStation,
    uint8_t      uSpsStation,
    uint8_t      uAutoInit,
    const char*  onlPwd,
    const bool   bSSL_TLS,
    SSL_VerifyServerCert_FUNCTYPE* pSSL_TLS_Callback
);
```

6.1.6.15 OnlineSSLH

Stellt eine Onlineverbindung zwischen dem PC (Projekt) und der Steuerung her.

```
extern "C" LSL_BOOL LASAL32_EXPORTS OnlineSSLH(
    int32_t      ocbNum,
    const char*  szComm,
    uint8_t      uBaudRate,
    uint8_t      uPcStation,
    uint8_t      uSpsStation,
    uint8_t      uAutoInit,
    const char*  onlPwd,
    const bool   bSSL_TLS,
```

```
SSL_VerifyServerCert_FUNCTYPE* pSSL_TLS_Callback
);
```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
szComm	const char*	Zeiger auf einen 0-terminierten String, der die Schnittstelle angibt, welche zum Aufbau der Verbindung verwendet wird. Wichtig: es muss der Port 2054 für eine SSL-Verbindung angegeben werden. Z.B.: "TCP:10.10.170.190:2054". Siehe auch OnlineH() .				
uBaudRate	uint8_t	Gibt die Baudrate an, mit der die RS232- oder CAN-BUS-Schnittstelle arbeitet oder ein Lese-Timeout für eine TCP/IP-Verbindung. Details siehe OnlineH() .				
uPcStation	uint8_t	Anzahl der CAN-Stationen an den PC (0-31). Für andere Anschlussarten ist dieser Parameter ungültig und sollte auf 0 gesetzt werden.				
uSpStation	uint8_t	Anzahl der CAN-Stationen der SPS (0-31). Für andere Anschlussarten ist dieser Parameter ungültig und sollte auf 0 gesetzt werden.				
uAutolnit	uint8_t	Parameter ist veraltet und sollte auf 0 gesetzt werden.				
onlPwd	const char*	Zeiger auf das Passwort für die Online-Verbindung. Dieser Parameter muss NULL sein, wenn die SPS kein Online-Passwort besitzt. Details siehe OnlinePwdH() .				
bSSL_TLS	const bool	TRUE: SSL/TLS soll verwendet werden				
pSSL_TLS_Callback	SSL_VerifyServerCert_FUNCTYPE	Zeiger auf eine Callback-Funktion des Benutzers, um das Zertifikat des Servers zu überprüfen. Dieser Parameter darf NULL sein, wenn bSSL_TLS==FALSE ist.				
Übergabeparameter	Typ	Beschreibung				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">TRUE</td><td>Erfolg</td></tr> <tr> <td>FALSE</td><td>Fehler</td></tr> </table> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die GetLastError()-Funktion auf.</p>	TRUE	Erfolg	FALSE	Fehler
TRUE	Erfolg					
FALSE	Fehler					

Prototyp der Callback-Funktion

```
LSL_BOOL _cdecl SSL_VerifyServerCert_FUNCTYPE(void*, uint32_t);
```

Übergabeparameter	Typ	Beschreibung
	void*	Zeiger auf SSvrCertInfo mit dem Server-Zertifikat

	uint32_t	Größe von SSvrCertInfo
Rückgabeparameter	Typ	Beschreibung
		TRUE Wenn der Benutzer dem Zertifikat vertraut
		FALSE Wenn der Benutzer dem Zertifikat nicht vertraut

Beispiel

```

char onlPwd[32] = { 0 };
const char* pPwd = onlPwd;

strcpy(onlPwd, "OnlinePassword");

if (OnlineSSL("TCP:10.10.170.190:2054", 0, 0, 0, 0, 0, pPwd, TRUE, OnlineSSLCallback))
{
    if (IsOnline())
    {
        if (LslReadDateTime(&SysTime))
        {
            printf("%d.%d.%d\n", SysTime.wDay, SysTime.wMonth, SysTime.wYear);
            printf("%d:%d:%d\n", SysTime.wHour, SysTime.wMinute, SysTime.wSecond);
        }
        else
            printf("LslReadDateTime() failed(0x%08X)\n", GetLastError());
        printf("\n");
    }
    else
        printf("OnlineSSL failed(0x%08X)\n", GetLastError());
}
else
    printf("OnlineSSL failed(0x%08X)\n", GetLastError());
Offline();

```

6.1.7 Modemfunktionen

Die Modemfunktionen können verwendet werden, um eine aktive Modemverbindung herzustellen. Um diese Funktionen nutzen zu können, muss das Modem zuerst im Betriebssystem mit den richtigen Parametern installiert werden. Weitere Informationen finden Sie in der LASAL Dokumentation Modemverbindung in Lasal.

6.1.7.1 ModemOpen

Die Funktion ModemOpen() initialisiert das Modem-Modul. Das Modem-Modul muss initialisiert werden, bevor die restlichen Modem-Funktionen aufgerufen werden können.

```
extern "C" LSL_BOOL LASAL32_EXPORTS ModemOpen(void);
```

Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich
		FALSE Fehler

6.1.7.2 ModemGetNumber

Die Funktion ModemGetNumber() liefert die Anzahl der installierten Modems zurück.

```
extern "C" DWORD LASAL32_EXPORTS ModemGetNumber(void);
```

Rückgabeparameter	Typ	Beschreibung
		Anzahl der verfügbaren Modems

6.1.7.3 ModemClose

Die Funktion ModemClose() deaktiviert eine aktive Verbindung.

```
extern "C" void WINAPI ModemClose(void);
```



Nachdem die Verbindung deaktiviert wurde, muss das Modem-Modul mit [ModemOpen\(\)](#) neu initialisiert werden, bevor die übrigen Modem-Funktionen aufgerufen werden können.

Beispiel

```
DWORD dModemNumber;
DWORD ModemSelected;
DWORD DialType;
char szTelNr[30];
char name[256];
int rc;
//
// Initialize modem
//
if ( ModemOpen() == FALSE )
```

```
{  
    printf( "ModemOpen failed !\n" );  
    return;  
}  
  
//  
//  Determine number of modems in the system  
//  
dModemNumber = ModemGetNumber();  
if ( dModemNumber < 1 )  
{  
    printf( "Number of modems == 0 !\n" );  
    ModemClose();  
    return;  
}  
//  
// Request all modem names  
//  
for( DWORD i = 0; i < dModemNumber; i++ )  
{  
    if ( ModemGetName( i, name, size of(name) ) == FALSE )  
    {  
        printf( "Name of the modem #%d could not be determined !\n", i );  
        ModemClose();  
        return;  
    }  
    printf( "Modem#%d = %s\n", i, name );  
}  
  
//  
//  Which of the available modems is to be used?  
//  
if ( dModemNumber > 1 )  
    while( 1 )  
    {  
        printf( "Which modem is to be used?" );  
  
        rc = scanf( "%ld", &ModemSelected );  
        if ( rc == 0 || rc == EOF )  
        {  
            printf( "**** Input error ***\n" );  
            continue;  
        }  
        if ( ModemSelected < 0 || ModemSelected >= dModemNumber )  
        {  
            printf( "**** Invalid modem number ***\n" );  
            continue;  
        }  
        break;  
    }  
else  
    ModemSelected = dModemNumber - 1;  
//  
// Request desired dialing procedure  
//
```

```
while( 1 )
{
    printf( "Dialing procedure (1=Puls, 0=Ton) ? " );

    rc = scanf( "%ld", &DialType );
    if ( rc == 0 || rc == EOF )
    {
        printf( "*** Input error ***\n" );
        continue;
    }
    if( DialType != 0 && DialType != 1 )
    {
        printf( "*** Invalid dialing procedure ***\n" );
        continue;
    }
    break;
}

// Request desired tel. number
// while( 1 )
{
    printf( "tel. no.? " );

    rc = scanf( "%s", szTelNr );
    if ( rc == 0 || rc == EOF )
    {
        printf( "*** Input error ***\n" );
        continue;
    }
    break;
}

printf( "Dial %s%s to Modem#%d...\n", DialType ? "P":"T",
                                                szTelNr,
                                                ModemSelected );
if ( ModemCall( ModemSelected,
                1,                      // 1 = Modem direct to the PLC
                DialType,    // "P"..pulse, "T"..tone
                szTelNr,
                100         // Timeout in sec.
                ) == FALSE )
{
    printf( "ModemCall failed !\n" );
    ModemClose();
    return -1;
}

printf( "O.K.\n" );

return;
```

6.1.7.4 ModemCall

Die Funktion ModemCall() startet den Wählvorgang.

```
extern "C" LSL_BOOL LASAL32_EXPORTS ModemCall(
    DWORD dModemID,
    BYTE uType,
    DWORD dConfigBaud,
    char *szTelNr,
    BYTE nTimeout
);
```

Übergabeparameter	Typ	Beschreibung	
dModemID	DWORD	0-basierter Index des Modems	
uType	BYTE	Legt den Verbindungstyp fest. Dieser Parameter muss auf 1 gesetzt werden (= direkter Verbindungstyp), da keine anderen Verbindungstypen an dieser Stelle implementiert werden.	
dConfigBaud	DWORD	Gibt an, ob Puls- oder Tonwahl verwendet wird	
		0 Tonwahl 1 Pulswahl	
szTelNr	CHAR	Zeiger auf einen 0-terminierten String, der die Telefonnummer enthält	
nTimeout	BYTE	Timeout, Wert in Millisekunden	
Übergabeparameter	Typ	Beschreibung	
		TRUE Funktion erfolgreich FALSE Fehler	

6.1.7.5 ModemGetName

Die Funktion ModemGetName() liefert eine Beschreibung eines bestimmten Modems zurück.

```
extern "C" LSL_BOOL LASAL32_EXPORTS ModemGetName(
    DWORD dModemID,
    char *pszName,
    WORD maxsize
);
```

Übergabeparameter	Typ	Beschreibung	

dModemID	DWORD	0-basierter Index des Modems
pszName	CHAR	Zeiger auf einen Charakterpuffer, der die Modem-Beschreibung enthält
maxsize	WORD	Definiert die Größe des Charakterpuffers pszname
Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich
		FALSE Fehler

6.1.8 Funktionen zum Datenaustausch

Datenaustausch mit Servern

LslGetObject() oder LslGetObjectEx() liefert die Kanaladresse des Servers.

Mit LslReadFromSrv() und LslWriteToSrv() kann auf einen numerischen Server zugegriffen werden.

Mit LslReadFromSrvStr() und LslWriteToSrvStr() kann auf einen String-Server zugegriffen werden.

Datenaustausch mit Clients

LslGetObject() oder LslGetObjectEx() liefert die Kanaladresse des Clients.

Mit LslReadFromClt() und LslWriteToClt() kann auf den Client zugegriffen werden.

Datenaustausch mit globalen Variablen

LslGetAdressVar() liefert die Adresse einer globalen Variablen.

Mit GetData(), LslGetDataEx() und SetData() kann dann auf die globale Variable zugegriffen werden.

Datenaustausch mit Objekt-Member-Variablen

Datenaustausch mit einer Speicheradresse auf der SPS

Auf Objekt-Member-Variablen einer Klasse oder auf mit malloc() allokierte Speicherbereiche kann nicht ohne weiteres zugegriffen werden, weil die Adressen nicht ermittelt werden können. Dieses Problem könnte wie folgt gelöst werden.

In der betreffenden Klasse werden ein Request-Server und ein Adressen-Server implementiert. Im Request-Server wird eingetragen, welche Adresse am Adressen-Server angezeigt werden soll.

Wenn nur auf eine oder wenige Adressen zugegriffen werden muss, kann auf den Request-Server verzichtet werden. Es wird dann nur ein Adressen-Server oder eine kleine Anzahl an Adressen-Servern implementiert, welche die gewünschten Adressen dauernd anzeigen. Mit `LslGetObject("Objektname.AdressenServername", ...)` und `LslReadFromSvr()` kann die angezeigte Adresse gelesen werden. Mit `GetData()`, `LslGetDataEx()` und `SetData()`, `LslSetDataEx()` kann dann auf die ermittelte Adresse zugegriffen werden.

Adresse eines Array-Elements

Die Adresse ergibt sich als Array-Basis-Adresse + Index * sizeof(Array-Element).

Adresse eines Struktur-Elements

Die Adresse ergibt sich als Struktur-Basis-Adresse + offsetof(StructType, Element).

Achtung: die Adresse ist nur dann korrekt, wenn das Alignment am PC und auf der SPS gleich ist

Austausch großer Datenmengen

Zum Lesen großer Datenmengen eignen sich die Funktionen `LslGetDataEx()`, `LslGetDataListEx()`, `LslGetDataListSpecial()` sowie `GetDataList()` zusammen mit `SendDataList()`.

Zum Schreiben großer Datenmengen eignet sich die Funktion `LslSetDataEx()`.

6.1.8.1 [GetData](#)

Entspricht der Funktion [GetDataH\(\)](#) mit `ocbNum = 0`.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetData(
    void        *pBuffer,
    uint32_t    addr0,
    uint16_t    nCount
);
```

6.1.8.2 GetDataH

Die Funktion GetDataH() ruft Daten aus der SPS ab.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetDataH(
    int32_t    ocbNum,
    void       *pBuffer,
    uint32_t   addr0,
    uint16_t   nCount
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
pBuffer	void	Zeiger auf den Speicherbereich, in dem die empfangenen Daten kopiert werden
addr0	uint32_t	Adresse des Datenbereichs in der SPS
nCount	uint16_t	Anzahl der zu erhaltenen Byte
Rückgabeparameter	Typ	Beschreibung
		<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>

Beispiel

```
BYTE pBuffer[1];
unsigned long Adr = 0x00401004;
WORD nCount = 1;

if ( GetDataH(0, (void*)pBuffer, Adr, nCount ) == TRUE )
    printf( "GetData( Adr=0x%08X ) = 0x%02X\n", Adr, pBuffer[0] );
else
    printf( "Error with GetData\n" );
```

6.1.8.3 GetDataList

Entspricht der Funktion [GetDataListH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetDataList(
    char   *data0
);
```

6.1.8.4 **GetDataListH**

Die Funktion **GetDataListH()** ruft Daten auf, deren Adressen in einem früheren Aufruf von **SendDataListH()** angegeben wurden.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetDataListH(
    int32_t    ocbNum,
    char       *data0
);
```

Übergabeparameter	Typ	Beschreibung		
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())		
data0	char	Zeiger auf einen Puffer, der die Datenliste abruf. Die Datenliste hat folgendes Format:		
		Offset	Typ	Beschreibung
		0	Unsigned short	Anzahl der Adressen
		2	Unsigned char	Daten Inhalt: Inhalt der 1.: Adresse, Inhalt der 2.: Adresse, usw.
		Die Adresse und Länge sind nicht in der Datenliste. Die Anwendung ist verantwortlich für die Zuteilung eines Puffers, der groß genug ist, um die Daten zu empfangen, deren Adressbereiche in einem Aufruf von SetDataList() festgelegt wurden.		
Übergabeparameter	Typ	Beschreibung		
		TRUE	Funktion erfolgreich	
		FALSE	Fehler	
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.				

Beispiel

```
char RetData[256], *pB;
unsigned short anz;

//
// Collect DataList data sent in the SendDataList example
//
if ( GetDataListH(0, RetData ) == TRUE )
{
    printf( "GetDataList O.K.\n" );
```

```

pB = RetData;

anz = *(unsigned short *)pB;
pB += sizeof(unsigned short);

printf( " Number of addresses: %d\n", anz );

if ( anz != 2 )
    printf( " Number of address incorrect !" );
else
{
    for ( int i = 0; i < 16; i++ )
        printf( "0x%02X ", (unsigned char)*pB++ );
    printf( "\n" );
}
else
    printf( "Error in GetDataList\n" );

```

6.1.8.5 LslExecKillH

Führt die Kill-Methode des vorgesehenen Servers aus.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslExecKillH(
    int32_t      ocbNum,
    uint32_t     dwSrvAddress,
    uint8_t      *pResult/*= NULL*/,
    uint32_t     dwResultCnt/*=0*/,
    uint32_t     useLenField,
    uint32_t     returnState
);

```

Übergabeparameter		Typ	Beschreibung
ocbNum	int32_t		Online Connection Block Nummer (Rückgabewert von OcbOpen())
dwSrvAddress	uint32_t		Zeiger auf den Server
pResult	uint8_t		Zeiger auf den Ergebnispuffer
dwResultCnt	uint32_t		Größe des Ergebnispuffers
useLenField	uint32_t		Anzahl der Ergebnisdaten im Längenbereich der Parameter
returnState	uint32_t		Auf TRUE setzen, um den Zustand der Methode namens kill abzufragen
Rückgabeparameter		Typ	Beschreibung
			TRUE Funktion erfolgreich
			FALSE Fehler

Beispiel

```
//Call the NewInst of a server
if( LslGetObjectH(0, "Class01", &addr, &mode, &clsname[ 0 ] ) )
{
m_output += "Object found: ";
memset( retbuf, 0xAA, 256 );
ret = LslExecNewInstrExH(0, addr, 0, ( unsigned long* )&parabuf[ 0 ],
2, ( unsigned char* )&retbuf[ 0 ], 256, FALSE, TRUE );
if( ret && *( unsigned long* )&retbuf[ 0 ] == 0x00000003 )
{
m_output += "NewInst returned BUSY\r\n";
SetTimer( 2, 10000, NULL );
}
else
{
if( ret == 0 )
m_output += "NewInst failed\r\n";
else
{
CString tmp;
tmp.Format( "NewInst returned %d\r\n",
*( unsigned long* )&retbuf[ 0 ] );
m_output += tmp;
}
}
}

// Timer2 routine
// cyclic call NewInst
ret = LslExecNewInstrExH(0, addr, 0, ( unsigned long* )&parabuf[ 0 ], 8,
( unsigned char* )&retbuf[ 0 ], 256, FALSE, TRUE );
if( ret && *( unsigned long* )&retbuf[ 0 ] == 0x00000000 )
{
m_output += "GetState returned READY\r\n";
KillTimer( 2 );
}
else
{
if( ret == 0 )
m_output += "GetState failed\r\n";
else
{
CString tmp;
tmp.Format( "GetState returned %d\r\n",
*( unsigned long* )&retbuf[ 0 ] );
m_output += tmp;
}
}

.
.
.
```

```
//if the NewInst will not get ready on its own, we have to kill it
ret = LslExecKillH(0, addr, (unsigned char*)&retbuf[ 0 ],
256, FALSE, TRUE );
if( ret )
{
m_output += "Killed\r\n";
KillTimer( 2 );
}
else
m_output += "Kill failed\r\n";
```

Der erste Aufruf von LslExecNewInstrExH() ruft die Methode NewInst der vorgesehenen Server auf. Je nach Stand der NewInst-Methode rufen nachfolgende Aufrufe entweder NewInst (wenn der vorherige Anruf abgeschlossen ist und die Funktion auf READY gesetzt ist) oder GetState (wenn der vorherige Aufruf noch verarbeitet wird und die Funktion auf BUSY gesetzt wurde) wieder auf. Eine Busy NewInst endet automatisch, wenn die Verbindung unterbrochen wird oder keine weiteren State-Abfragen vorgenommen werden. Um eine BUSY NewInst zu beenden, kann der Benutzer den Aufruf der LslExecKill-Funktion verwenden. Dieser Aufruf wird die Servermethode beenden und den Zustand wieder auf READY setzen. Mehrere Aufrufe zu einer einzelnen NewInst-Methode werden nicht unterstützt (nachfolgende Aufrufe würden die GetState-Funktion auslösen).

6.1.8.6 LslExecNewInstrEx

Entspricht der Funktion [LslExecNewInstrExH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslExecNewInstrEx(
    WORD          dwSrvAddress,
    WORD          wCmd,
    DWORD         adwParas[],
    DWORD         dwParaCnt,
    unsigned char *pResult,
    DWORD         dwResultCnt,
    DWORD         useLenField,
    DWORD         returnState
);
```

6.1.8.7 LslExecNewInstrExH

Führt die NewInst/GetState Methode des bezeichneten Servers aus.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslExecNewInstrExH(
    int32_t        ocbNum,
    DWORD          dwSrvAddress,
    WORD           wCmd,
    DWORD          adwParas[],
```

```

    DWORD      dwParaCnt,
    unsigned char *pResult,
    DWORD      dwResultCnt,
    DWORD      useLenField,
    DWORD      returnState
);

```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
dwSrvAddress	DWORD	Zeiger auf den Server	
wCmd	WORD	Befehl zur Ausführung durch die NewInst/GetState Methode	
adwParas[]	DWORD	Array von Parametern der Methode (4 Byte pro Parameter)	
dwParaCnr	DWORD	Anzahl der Parameter im obenstehenden Array. Es sind maximal 20 Parameter möglich.	
pResult	UNSIGNED CHAR	Zeiger auf den Ergebnispuffer oder NIL, wenn keine Rückgabedaten benötigt werden. Das erste WORD im Puffer enthält die Länge der Daten inkl. dieses Längenfelds. Die maximale Länge ist 252 Byte.	
dwResultCnt	DWORD	Anzahl der Byte, die von der Rückmeldung erwartet werden (useLenField = 0) oder Größe des Ergebnispuffers (useLenField ≠ 0).	
useLenField	DWORD	0: Es werden mindestens dwResultCnt Byte von der Rückmeldung erwartet. Sind weniger Daten vorhanden, dann scheitert die Funktion. Wenn mehr Daten vorhanden sind, dann werden die überflüssigen Daten verworfen. #0: Es wird versucht, dem Aufrufer den gesamten Ergebnispuffer zu übergeben. D.h. das Längenfeld in der Rückmeldung wird interpretiert und nur die darin definierte Anzahl von Byte wird in den Ergebnispuffer kopiert. dwResultCnt ist die Größe des Puffers des Aufrufers. Wenn dwResultCnt zu klein ist, scheitert die Funktion.	
returnState	DWORD	Auf TRUE setzen, um den Zustand der aufgerufenen NewInst/GetState Methode abzufragen. Der Zustand wird im ersten DWORD des Ergebnispuffers zurückgeliefert.	
Rückgabeparameter	Typ	Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler

Mit Befehlsaufrufen können Objektbefehle ausgelöst werden. Die Befehle werden in der NewInstr-Methode der Klasse durchgeführt. Befehlsaufrufe werden oft zur Datenübertragung verwendet.

6.1.8.8 LslGetAdressVar

Entspricht der Funktion [LslGetAdressVarH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetAdressVar(
    const char    *name,
    int32_t        *adresse
);
```

6.1.8.9 LslGetAdressVarH

Die Funktion LslGetAdressVarH() bestimmt die Adresse einer globalen Variable in der SPS.

Mit den Funktionen LslGetObject() bzw. LslGetObjectEx() können die Adressen von Klassenobjekten und von Clients und Servern ermittelt werden.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetAdressVarH(
    int32_t        ocbNum,
    const char    *name,
    int32_t        *adresse
);
```

Diese Funktion kann die Adresse eines Klassenobjekts, einer Funktion oder eines Elements in einer Struktur nicht ermitteln.



Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
name	CONST CHAR	Zeiger auf einen 0-terminierten String, der den Namen der Variable enthält
Address	int32_t	Legt einen Zeiger auf eine Variable fest, der die Adresse der globalen Variable erhält. Wenn dieser Parameter ein Wert von 0 erhält, konnte die Adresse der globalen Variable nicht ermittelt werden.
Rückgabeparameter	Typ	Beschreibung
		<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>

Beispiel

```

unsigned long address;
char *name = "OPS";

if ((LslGetAddressVar( name, (long*)&address ) == TRUE) &&
    (address != 0) )
    printf( "Address of %s = 0x%08X\n", name, address );
else
    printf( "Error in LslGetAddressVar\n" );

```

6.1.8.10 LslGetDataEx

Entspricht der Funktion [LslGetDataExH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDataEx(
    uint8_t    *pBuffer,
    uint32_t   addr0,
    uint32_t   nCount
);

```

6.1.8.11 LslGetDataExH

Die Funktion LslGetDataExH() ruft Daten von der SPS ab (über 32000 Byte können gelesen werden)

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDataExH(
    int32_t    ocbNum,
    uint8_t    *pBuffer,
    uint32_t   addr0,
    uint32_t   nCount
);

```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
pBuffer	uint8_t	Zeiger auf den Speicherbereich, in dem die empfangenen Daten kopiert werden	
addr0	uint32_t	Adresse der Daten im Zielsystem	
nCount	uint32_t	Anzahl der zu erhaltenen Bytes	
Übergabeparameter	Typ	Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler

		Mit LslGetError kann der letzter Fehler-Code ermittelt werden. Dieser Code entspricht entweder einem Lasal32-Fehler-Code oder einem systemspezifischen Fehler-Code (Windows: GetLastError() oder Linux: errno).
--	--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Beispiel

```
BYTE pBuffer[1];
unsigned long Adr = 0x00401004;
DWORD nCount = 1;

if ( LslGetDataExH(0, (void*)pBuffer, Adr, nCount ) == TRUE )
    printf( "LslGetDataEx( Adr=0x%08X ) = 0x%02X\n", Adr, pBuffer[0] );
else
    printf( "Error with LslGetDataEx\n" );
```

6.1.8.12 LslGetDataListEx

Entspricht LslGetDataListExH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDataListEx(
    const DWORD      dwCount,
    const DWORD      *pAddr,
    const DWORD      *pLen,
    unsigned char    *pResult
);
```

6.1.8.13 LslGetDataListExH

Diese Funktion liest mehrere Daten von der SPS. Sie versucht dabei das Optimum an Geschwindigkeit zu erzielen. Dazu werden die Funktionen SendDataList() / GetDataList() / GetData() verwendet.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDataListExH(
    int32_t          ocbNum,
    const DWORD      dwCount,
    const DWORD      *pAddr,
    const DWORD      *pLen,
    unsigned char    *pResult
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
dwCount	const DWORD	Anzahl der Kommandos
pAddr	const DWORD*	Liste der SPS-Adressen, von wo gelesen werden soll

pLen	const DWORD*	Liste mit den Datenlängen der Ergebnisse
pResult	char*	Liste mit den Ergebnissen
Rückgabeparameter	Typ	Beschreibung
		<div style="display: flex; justify-content: space-between; align-items: center;"> TRUE Funktion erfolgreich </div> <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 10px;"> FALSE Fehler </div>
		Um erweiterte Fehlerinformationen zu erhalten, rufen Sie die Funktion GetLastError() auf.

Beispiel

```

static const int iMax = 200;
BOOL rs = FALSE;
uint8_t buffer[100000];
uint32_t dwAddr[iMax];
uint32_t dwLen[iMax];

rs = LslGetAdressVar("g_BigBuf", dwAddr);
if (rs == TRUE && dwAddr[0])
{
    //Gültige Adresse in dwAddr[0]
    //Gültige Länge in dwLen[0]
    dwLen[0] = 4;

    for(int i = 1; i < iMax; i++)
    {
        //Jedes zweite Element im Puffer "g_BigBuf" lesen
        dwAddr[i] = dwAddr[0] + i*2*4;
        dwLen[i] = 4;
    }

    memset(buffer, 0, sizeof(buffer));
    rs = LslGetDataListEx(iMax, dwAddr, dwLen, (uint8_t*)buffer);
    if (rs == FALSE)
    {
        printf("LslGetDataListEx() failed(0x%08X)\n", GetLastError());
    }
    else
    {
        printf("LslGetDataListEx() OK, DataLen=%d\n", strlen(buffer));
        printf("Buffer[] = %s\n", buffer);
    }
}
else
    printf("LslGetAdressVar failed(0x%08X)\n", GetLastError());

```

6.1.8.14 LslGet dataListSpecial

Entspricht LslGet dataListSpecialH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGet dataListSpecial(
    const DWORD dwCount,
    const DWORD *pAddr,
    const DWORD *pLen,
    char *pResult
);
```

6.1.8.15 LslGet dataListSpecialH

Diese Funktion führt mehrere GetData Kommandos in einem aus.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGet dataListSpecialH(
    int32_t ocbNum,
    const DWORD dwCount,
    const DWORD *pAddr,
    const DWORD *pLen,
    char *pResult
);
```

Übergabeparameter		Typ	Beschreibung
ocbNum		int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
dwCount		const DWORD	Anzahl der Kommandos
pAddr		const DWORD*	Liste der SPS-Adressen, von wo gelesen werden soll
pLen		const DWORD*	Liste mit den Datenlängen der Ergebnisse
pResult		char*	Liste mit den Ergebnissen
Rückgabeparameter		Typ	Beschreibung
			<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p>
			Um erweiterte Fehlerinformationen zu erhalten, rufen Sie die Funktion GetLastError() auf.

Beispiel

```
static const int iMax = 200;
BOOL rs = FALSE;
uint8_t buffer[100000];
uint32_t dwAddr[iMax];
uint32_t dwLen[iMax];
```

```

rs = LslGetAddressVar("g_BigBuf", dwAddr);
if (rs == TRUE && dwAddr[0])
{
    //Gültige Adresse in dwAddr[0]
    //Gültige Länge in dwLen[0]
    dwLen[0] = 4;

    for(int i = 1; i < iMax; i++)
    {
        //Jedes zweite Element im Puffer "g_BigBuf" lesen
        dwAddr[i] = dwAddr[0] + i*2*4;
        dwLen[i] = 4;
    }

    memset(buffer, 0, sizeof(buffer));
    rs = LslGetDataListSpecial(iMax, dwAddr, dwLen, buffer);
    if (rs == FALSE)
    {
        printf("LslGetDataListSpecial() failed(0x%08X)\n", GetLastError());
    }
    else
    {
        printf("LslGetDataListSpecial() OK, DataLen=%d\n", strlen(buffer));
        printf("Buffer[] = %s\n", buffer);
    }
}
else
    printf("LslGetAddressVar failed(0x%08X)\n", GetLastError());

```

6.1.8.16 LslSetDataEx

Entspricht LslSetDataExH() mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslSetDataEx(
    const uint8_t* pBuffer,
    DWORD          addr0,
    DWORD          nCount
);

```

6.1.8.17 LslSetDataExH

Diese Funktion sendet Daten an die SPS wie SetData(), aber ohne Längenbeschränkung.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslSetDataExH(
    int32_t        ocbNum,
    const uint8_t* pBuffer,
    DWORD          addr0,
    DWORD          nCount
);

```

Übergabeparameter	Typ	Beschreibung
-------------------	-----	--------------

ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
pBuffer	const uint8_t*	Zeiger auf die Daten, die gesendet werden sollen
Addr0	DWORD	Adresse in der SPS
nCount	DWORD	Anzahl der zu sendenden Byte
Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich
		FALSE Fehler
		Um erweiterte Fehlerinformationen zu erhalten, rufen Sie die Funktion GetLastError() auf.

Beispiel

```

BOOL bResult = FALSE;
plcptr_t nObjAddr = 0;
uint32_t nCount = 100000;
uint8_t buffer[100000];

bResult = LslGetAdressVar("g_BigBuf", &nObjAddr);
if (bResult == TRUE && nObjAddr)
{
    //Gültige Adresse in nObjAddr
    //ToDo:Puffer füllen

    bResult = LslSetDataEx(buffer, nObjAddr, nCount));
    if (bResult == FALSE)
        printf("LslSetDataEx failed(0x%08X)\n", GetLastError());
}
else
    printf("LslGetAdressVar failed(0x%08X)\n", GetLastError());
    
```

6.1.8.18 SendDataList

Entspricht der Funktion [SendDataListH\(\)](#) mit ocbNum = 0

```

extern "C" LSL_BOOL LASAL32_EXPORTS SendDataList(
    uint16_t      length0,
    char          *data0,
    CB_FUNCTYPE   *callback
);
    
```

6.1.8.19 SendDataListH

Die Funktion SendDataListH() schickt eine Liste von Adressen an die SPS, deren Inhalt später gesammelt über die Funktion [Get dataList\(\)](#) verwendet werden kann.

```
typedef unsigned long _cdecl CB_FUNCTYPE(unsigned long);

extern "C" LSL_BOOL LASAL32_EXPORTS SendDataListH(
    int32_t      ocbNum,
    uint16_t      length0,
    char*         *data0,
    CB_FUNCTYPE*  *callback
);
```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
Length0	uint16_t	Anzahl der Byte in data0				
data0	char*	Zeiger auf einen Puffer, der die Liste der Adressen enthält. Die Liste hat folgendes Format:				
		Offse	Typ			
		t	Beschreibung			
		0	Unsigned short			
		2	Unsigned long			
		6	Unsigned short			
		8	Unsigned long			
callback	CB_FUNCTYPE*	12	Unsigned short			
		Anzahl der Bytes in der 2. Adresse				
Und so weiter						
Rückgabeparameter	Typ	Beschreibung				
	LSL_BOOL	TRUE	Funktion erfolgreich			
		FALSE	Fehler			

		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.
--	--	---------------------------------------------------------------------------------------------------------

Beispiel

```
char DataList[256], *pB;
unsigned short len = 0;

pB = DataList;

// Anzahl der Adressen 2
*(unsigned short *)pB = 2;
pB += sizeof(unsigned short);

// 1st.Address: 0x00401004
*(unsigned long *)pB = 0x00401004;
pB += sizeof(unsigned long);

// Anzahl der Bytes an der 1. Adresse 4
*(unsigned short *)pB = 4;
pB += sizeof(unsigned short);

// 2nd address: 0x00401008
*(unsigned long *)pB = 0x00401008;
pB += sizeof(unsigned long);

// Anzahl der Bytes an der 2. Adresse 4
*(unsigned short *)pB = 4;
pB += sizeof(unsigned short);

if ( SendDataListH(0, pB-DataList, DataList, ZERO/*callback*/ ) == TRUE )
    printf( "SendDataList O.K.\n" );
else
    printf( "Error in SendDataList\n" );
```

6.1.8.20 SetData

Entspricht der Funktion [SetDataH\(\)](#) mit ocbNum = 0

```
extern "C" LSL_BOOL LASAL32_EXPORTS SetData(
    const void    *pBuffer,
    uint32_t      addr0,
    uint16_t      nCount
);
```

6.1.8.21 SetDataH

Die Funktion SetDataH() sendet Daten an die SPS.

```
extern "C"  LSL_BOOL LASAL32_EXPORTS SetDataH(
    int32_t      ocbNum,
    const void  *pBuffer,
    uint32_t     addr0,
    uint16_t     nCount
);
```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
pBuffer	CONST VOID	Zeiger auf die Daten, die gesendet werden sollen	
addr0	uint32_t	Adresse in der SPS	
nCount	uint16_t	Anzahl von zu sendenden Byte	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.			

Beispiel

```
BYTE pBuffer[1];
unsigned long Adr = 0x00401004;
WORD nCount = 1;

pBuffer[0] = 0xAC;

if ( SetDataH(0, (void*)pBuffer, Adr, nCount ) == TRUE )
    printf( "SetData( Adr=0x%08X, Data=0x%02X ) o.k.\n", Adr,
            pBuffer[0] );
else
    printf( "Error in SetData\n" );
```

6.1.9 Status-Informationen und SPS-Befehle

6.1.9.1 GetBusType

Entspricht GetBusTypeH() mit ocbNum = 0.

```
extern "C" uint32_t LASAL32_EXPORTS GetBusType(void);
```

6.1.9.2 GetBusTypeH

Gibt die die momentan eingestellte Kommunikationsart zurück.

```
extern "C" uint32_t LASAL32_EXPORTS GetBusTypeH(
    int32_t    ocbNum
);
```

Übergabeparameter		Typ	Beschreibung	
Rückgabeparameter		Typ	Beschreibung	
		Wert	Name	Beschreibung
		0	COMMUNICATION_NONE	Keine Kommunikation
		1	COMMUNICATION_V24	Kommunikation über V24
		2	COMMUNICATION_CAN	Kommunikation über CAN-Bus
		3	COMMUNICATION_TCP	Kommunikation über TCP/IP
		4	COMMUNICATION_MODEM	Kommunikation über Modem
		5	COMMUNICATION_CAN_USB	Kommunikation über das Lawicel CANUSB Interface
		6	COMMUNICATION_CAN_USB_FT	Kommunikation über das Lawicel CANUSB Interface (FTD2XX Treiber)
		7	COMMUNICATION_VARAN32	Kommunikation über VARAN32.DLL

Beispiel

```
if (GetBusType() == COMMUNICATION_TCP)
{
```

```

    //Communication via TCP/IP
}

```

6.1.9.3 GetChk

Entspricht GetChkH() mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS GetChk(
    void* pChk
);

```

6.1.9.4 GetChkByType

Entspricht GetChkByTypeH() mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS GetChkByType(
    void* pChk,
    uint32_t modTypeMask,
    uint32_t modTypeEq
);

```

6.1.9.5 GetChkByTypeH

Die Funktion liefert die Prüfsumme des Projekts. Bei der Berechnung werden nur jene Module berücksichtigt, die mit modTypeMask UND-verknüpft den Wert modTypeEq ergeben. Die Prüfsumme ist die Summe der Modul-CRCs. Der Startwert ist 0, d.h. wenn kein Modul vorhanden ist, ist die Prüfsumme 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS GetChkByTypeH(
    int32_t ocbNum,
    void* pChk,
    uint32_t modTypeMask,
    uint32_t modTypeEq
);

```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
pChk	void*	Zeiger auf eine Variable, wo die Prüfsumme des Projekts eingetragen wird (uint32_t)
modTypeMask	uint32_t	Maske für den Modul-Typ
modTypeEq	uint32_t	Nur dieser Modul-Typ wird berücksichtigt
Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich

		FALSE Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.		

Die folgenden defines können zur Auswahl der Module verwendet werden.

```
#define LSLMODTYPE_LOADER      0x00000001
#define LSLMODTYPE_APPL        0x00010000
```

Beispiel 1 (Prüfsumme über alle Module berechnen)

```
uint32_t CheckSum=0;
uint32_t modTypeMask=0;
uint32_t modTypeEq=0;

if (GetChkByType(&CheckSum, modTypeMask, modTypeEq))
    printf("CheckSum=%d\n", CheckSum);
else
    printf("GetChkByType() failed(0x%08X)\n", GetLastError());
```

Beispiel 2 (Prüfsumme nur über Applikationsmodule berechnen)

```
uint32_t CheckSum=0;
uint32_t modTypeMask=LSLMODTYPE_APPL;
uint32_t modTypeEq=LSLMODTYPE_APPL;

if (GetChkByType(&CheckSum, modTypeMask, modTypeEq))
    printf("CheckSum=%d\n", CheckSum);
else
    printf("GetChkByType() failed(0x%08X)\n", GetLastError());
```

6.1.9.6 GetChkH

Die Funktion liefert die Prüfsumme des Projekts. Bei der Berechnung werden nur jene Module berücksichtigt, die mit der Voreinstellung von modTypeMask UND-verknüpft den voreingestellten Wert von modTypeEq ergeben. Die Voreinstellungen werden mit den Funktionen LslSetDfltModTypeParams() bzw. LslSetDfltModTypeParamsH() gesetzt.

Die Prüfsumme ist die Summe der Modul-CRCs. Der Startwert ist 0, d.h. wenn kein Modul vorhanden ist, ist die Prüfsumme 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetChkH(
    int32_t    ocbNum,
    void*      pChk
);
```

Übergabeparameter	Typ	Beschreibung

ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
pChk	void*	Zeiger auf eine Variable, wo die Prüfsumme des Projekts eingetragen wird (uint32_t)				
Rückgabeparameter	Typ	Beschreibung				
		<table border="1" style="width: 100px; margin-left: auto; margin-right: auto;"> <tr> <td style="width: 50px; text-align: center;">TRUE</td><td>Funktion erfolgreich</td></tr> <tr> <td style="width: 50px; text-align: center;">FALSE</td><td>Fehler</td></tr> </table>	TRUE	Funktion erfolgreich	FALSE	Fehler
TRUE	Funktion erfolgreich					
FALSE	Fehler					
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.				

Beispiel

```
uint32_t CheckSum=0;

if (GetChk(&CheckSum))
    printf("CheckSum=%d\n", CheckSum);
else
    printf("CheckSum() failed(0x%08X)\n", GetLastError());
```

6.1.9.7 GetCpuStatus

Entspricht der Funktion [GetCpuStatusH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetCpuStatus(
    uint8_t *pStatus
);
```

6.1.9.8 GetCpuStatus2

Entspricht GetCpuStatus2H() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetCpuStatus2(
    uint8_t *pStatus,
    uint8_t direct
);
```

6.1.9.9 GetCpuStatus2H

Die Funktion fragt den CPU-Status ab.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetCpuStatus2H(
    int32_t ocbNum,
    uint8_t *pStatus,
    uint8_t direct
);
```

Übergabeparameter		Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
pStatus	uint8_t*	Zeiger auf eine Byte-Variable, die die CPU-Statuscodes der SPS erhält. Eine Liste der SPS-Statuscodes finden Sie im Anhang A .	
direct	uint8_t	0	Es wird möglicherweise ein gepufferter (= veralteter) Status zurückgeliefert
		≥0	Es wird der aktuelle Status beschafft
Rückgabeparameter		Typ	Beschreibung
		TRUE	Funktion erfolgreich
		FALSE	Fehler
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	

Beispiel

```

BOOL bOk = false;
uint8_t byCpuState = 39;    // 39 = CS_OFFLINE
bool bCpuStateOk = FALSE;

// Hole aktuellen CPU Status
bOk = GetCpuStatus2(&byCpuState, 1);
if (bOk)
{
    if (byCpuState == CS_RUN_RAM || byCpuState == CS_RUN_ROM)
    {
        bCpuStateOk = true;
    }
}

```

6.1.9.10 GetCpuStatusH

Die Funktion GetCpuStatusH() ruft die CPU-Statuscodes der PLC auf.

```

extern "C" LSL_BOOL LASAL32_EXPORTS GetCpuStatusH(
    int32_t ocbNum,
    uint8_t *pStatus
);

```

Übergabeparameter		Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	

Pstatus	uint8_t	Zeiger auf eine Byte-Variable, der die CPU-Statuscodes der PLC erhält. Eine Liste der CPU-Statuscodes finden Sie im Anhang A .
Rückgabeparameter	Typ	Beschreibung
		<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>

Beispiel

```
BYTE Status;

if ( GetCpuStatus( &Status ) == TRUE )
    printf( "CpuStatus = 0x%02X\n", Status );
else
    printf( "Error in GetCpuStatus\n" );
```

6.1.9.11 GetPowerFailInfo

Die Funktion ist veraltet.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetPowerFailInfo(
    uint8_t    *info,
    uint8_t    *errcode
);
```

Übergabeparameter	Typ	Beschreibung								
info	uint8_t	<p>Zeiger auf eine Variable, wo eingetragen wird, ob ein PowerFail aufgetreten ist.</p> <table> <tr> <td>0</td><td>Kein PowerFail aufgetreten oder PowerFailInfo konnte nicht ermittelt werden (rc=FALSE)</td></tr> <tr> <td>1</td><td>PowerFail ist aufgetreten</td></tr> </table>	0	Kein PowerFail aufgetreten oder PowerFailInfo konnte nicht ermittelt werden (rc=FALSE)	1	PowerFail ist aufgetreten				
0	Kein PowerFail aufgetreten oder PowerFailInfo konnte nicht ermittelt werden (rc=FALSE)									
1	PowerFail ist aufgetreten									
errcode	uint8_t	<p>Zeiger auf eine Variable, in der die Fehlerursache eingetragen wird, wenn keine PowerFail-Info ermittelt werden konnte (rc=FALSE).</p> <table> <tr> <td>0</td><td>Kein Fehler</td></tr> <tr> <td>1</td><td>SIGMA-Treiber konnte nicht geöffnet werden</td></tr> <tr> <td>2</td><td>Falsche Version des SIGMA-Treibers</td></tr> <tr> <td>3</td><td>Kein SIGMATEK-PC</td></tr> </table>	0	Kein Fehler	1	SIGMA-Treiber konnte nicht geöffnet werden	2	Falsche Version des SIGMA-Treibers	3	Kein SIGMATEK-PC
0	Kein Fehler									
1	SIGMA-Treiber konnte nicht geöffnet werden									
2	Falsche Version des SIGMA-Treibers									
3	Kein SIGMATEK-PC									

		4 UPS ist nicht enabled
		99 System error (out of memory, system-call failed, ...)
Rückgabeparameter	Typ	Beschreibung
		TRUE PowerFail-Info konnte ermittelt werden
		FALSE PowerFail-Info konnte nicht ermittelt werden

Beispiel

```
uint8_t info = 0;
uint8_t errcode = 0;

if (GetPowerFailInfo(&info, &errcode))
{
    if (info)
        printf("Power Fail occurred\n");
    else
        printf("No Power Fail occurred\n");
}
else
    printf("GetPowerFailInfo() failed(%d)\n", errcode);
```

6.1.9.12 LslGetDllInfo

Gibt einen Zeiger auf einen ASCII-0-String zurück. Der String enthält detaillierte Angaben zur Lasal32.dll.

```
extern "C" const char* LASAL32_EXPORTS LslGetDllInfo(void);
```

Rückgabewert

```
<DLLInfo>
  <DLL Name=Lasal32
    Version=01.01.149
    Build="dev"
    Platform="Win32"
    CompileDate="Jul 11 2023"
    CompileTime="11:18:57"
    DBG32LEVEL="0"
    DBGRLLEVEL="0"
  </DLLInfo>
```

DBG32LEVEL: Debug-Level der Lasal32.dll

DBGRLLEVEL: Debug-Level der Refresh-Liste

Beispiel

```
const char *pInfo = LslGetDllInfo();
printf("DLLInfo: %s\n", pInfo);
```

6.1.9.13 LslGetDllVersion

Gibt die DLL-Versionsnummer zurück.

```
extern "C" uint32_t LASAL32_EXPORTS LslGetDllVersion(void);
```

Rückgabewert

Die Versionsnummer hat folgenden Aufbau:

HIGH-WORD				LOW-WORD			
31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
		Major		Minor		Subversion	

Beispiel

```
unsigned long DllVersion = LslGetDllVersion();
printf("DLL-Version: %d.%d.%d\n", DllVersion>>12 & 0x0000000F, DllVersion>>8 & 0x0000000F, DllVersion & 0x0000000F);
```

Die DLL-Version 1.1.149 entspricht dem Hex-Wert 16#00001195.

6.1.9.14 LslGetMemInfo

Entspricht LslGetMemInfoH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetMemInfo(
    uint8_t* pData,
    uint32_t len
);
```

6.1.9.15 LslGetMemInfoH

Liefert Startadresse, Länge, benutzte Größe und Attribut von Speicherbereichen der SPS wie Codebereich, RAM Bereich, SRAM Bereich. Ein Längenwert < 0 ist ein Fehler-Code.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetMemInfoH(
    int32_t    ocbNum,
    uint8_t*   pData,
    uint32_t   len
);
```

Übergabeparameter		Typ	Beschreibung	
ocbNum		int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
pData		uint8_t*	Zeiger auf den Ergebnispuffer	
len		uint32_t	Länge des Ergebnispuffers in Byte	
Rückgabeparameter		Typ	Beschreibung	
			TRUE	Funktion erfolgreich
			FALSE	Fehler
			Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	

Die ersten vier Byte von pData enthalten die Anzahl der Elemente des nachfolgenden Arrays von Strukturen (1 Element pro Speicherbereich). Der Datentyp der Strukturen ist S_MEM_INFO. Das Element attrib gibt an, ob die Elemente start, size und used gültig sind oder nicht.

```

typedef enum
{
    MI_CODE,           // 0 - code area
    MI_DATA,           // 1 - data area
    MI_OS_HEAP,        // 2 - os heap
    MI_USER_HEAP,      // 3 - user heap
    MI_SRAM_USER,      // 4 - sram user area
    MI_SRAM_RETAIN,    // 5 - sram retain area
    MI_SRAM_RET_RAM,   // 6 - sram retain ram area
    MI_SRAM_SYS,        // 7 - sram system area (Kernel-Log,...)
    MI_SRAM_COMPL,     // 8 - sram complete area (user + retain + system)
} E_MEM_INFO_TYPE;

typedef struct
{
    E_MEM_INFO_TYPE    type;
    uint32_t           start;           // start address
    uint32_t           size;            // length of memory area
    uint32_t           used;            // used data length
    uint32_t           attrib;          // attribute of the memory area;
} S_MEM_INFO;

#define MI_ATTRIB_START 0x0001 // struct member start is valid
#define MI_ATTRIB_LENGTH 0x0002 // struct member length is valid
#define MI_ATTRIB_USED   0x0004 // struct member used is valid

```

Beispiel

```

int32_t MemInfoCount = 0;
S_MEM_INFO* pMemInfo = NULL;

```

```
int i;

if (LslGetMemInfo((uint8_t*)&buffer, sizeof(buffer)))
{
    MemInfoCount = *(int32_t*)buffer;      //error code (<0) or number of S_MEM_INFO-Elements
    pMemInfo = (S_MEM_INFO*)(buffer + sizeof(MemInfoCount));
    if (MemInfoCount < 0)
    {
        printf("LslGetMemInfo() returned error code: %d\n", MemInfoCount);
    }
    else
    {
        for (i = 0; i < MemInfoCount; i++)
        {
            switch (pMemInfo->type)
            {
                case MI_CODE:
                    printf("===== Code area =====\n");
                    break;

                case MI_DATA:
                    printf("===== Data area =====\n");
                    break;

                case MI_OS_HEAP:
                    printf("===== OS heap =====\n");
                    break;

                case MI_USER_HEAP:
                    printf("===== User heap =====\n");
                    break;

                case MI_SRAM_USER:
                    printf("===== SRAM user area =====\n");
                    break;

                case MI_SRAM_RETAIN:
                    printf("===== SRAM retain area =====\n");
                    break;

                case MI_SRAM_RET_RAM:
                    printf("===== SRAM retain RAM area =====\n");
                    break;

                case MI_SRAM_SYS:
                    printf("===== SRAM system area (Kernel-Log,...) =====\n");
                    break;

                case MI_SRAM_COMPL:
                    printf("===== SRAM complete area (user + retain + system) =====\n");
                    break;
            }
        }

        if (pMemInfo->attrib & MI_ATTRIB_START)
            printf("Start address: 0x%08X\n", pMemInfo->start);
    }
}
```

```

        if (pMemInfo->attrib & MI_ATTRIB_LENGTH)
            printf("Length: %d\n", pMemInfo->size);
        if (pMemInfo->attrib & MI_ATTRIB_USED)
            printf("Used length: %d\n", pMemInfo->used);

        printf("\n");
        pMemInfo++;
    }
}
else
    printf("LslGetMemInfo() failed(0x%08X)\n", GetLastError());

```

6.1.9.16 LslGetPLCInfo

Entspricht der Funktion [LslGetPLCInfoH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetPLCInfo(
    uint8_t    *infostr,
    uint32_t   maxsize0
);

```

6.1.9.17 LslGetPLCInfoH

Die Funktion LslGetPLCInfoH() ruft allgemeine Informationen aus der SPS auf, z.B. Versionsnummer des Betriebssystems, Adressen der Anwender-Daten und Anwender-Code Bereich.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetPLCInfoH(
    int32_t      ocbNum,
    unsigned char *infostr,
    unsigned long maxsize0
);

```

Übergabeparameter	Typ	Beschreibung															
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())															
infostr		<p>Zeiger auf einen Puffer, der die SPS-Informationen empfängt. Die SPS-Informationen haben folgendes Format:</p> <table border="1"> <thead> <tr> <th>Offset</th><th>Typ</th><th>Beschreibung</th></tr> </thead> <tbody> <tr> <td>0</td><td>Unsigned long</td><td>Benutzer-Code Startadresse</td></tr> <tr> <td>4</td><td>Unsigned long</td><td>Benutzer-Code Länge</td></tr> <tr> <td>8</td><td>Unsigned long</td><td>Benutzer-Daten Startadresse</td></tr> <tr> <td>12</td><td>Unsigned long</td><td>Benutzer-Daten Länge</td></tr> </tbody> </table>	Offset	Typ	Beschreibung	0	Unsigned long	Benutzer-Code Startadresse	4	Unsigned long	Benutzer-Code Länge	8	Unsigned long	Benutzer-Daten Startadresse	12	Unsigned long	Benutzer-Daten Länge
Offset	Typ	Beschreibung															
0	Unsigned long	Benutzer-Code Startadresse															
4	Unsigned long	Benutzer-Code Länge															
8	Unsigned long	Benutzer-Daten Startadresse															
12	Unsigned long	Benutzer-Daten Länge															

		16	Unsigned long	(Nur intern verwendet)
		20	0-terminierter String	PLC-Name
		..	0-terminierter String	Grafikmodus
		..	0-terminierter String	Version
		..	0-terminierter String	Bootimage-Typ ("OSBIN" = 386, "OSRTB" = IPCs)
		..	0-terminierter String	"LSE", wenn CPU IPC-Grafik hat und vom LSE-Kernel unterstützt wird. 3 Leerzeichen, wenn LSE nicht unterstützt wird.
Zusätzliche Informationen, je nach Betriebssystem-Version, nur für die interne Nutzung bleiben undokumentiert.				
maxsize0		Gibt die Größe der SPS-Informationspuffer in Byte an. Die Größe des Puffers sollte groß genug sein (> 100 Byte).		
Rückgabeparameter	Typ	Beschreibung		
		TRUE	Funktion erfolgreich	
		FALSE	Fehler	
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.				

0-terminierter String <Grafikmodus>

```

G0  Textmode
G10 Graphicmode 320x200x16
G11 Graphicmode 320x200x256
G12 Graphicmode 320x200x64k
G20 Graphicmode 640x480x16
G21 Graphicmode 640x480x256
G22 Graphicmode 640x480x64k
G30 Graphicmode 800x600x16
G31 Graphicmode 800x600x256
G32 Graphicmode 800x600x64k
G40 Graphicmode 1024x768x16
G41 Graphicmode 1024x768x256
G42 Graphicmode 1024x768x64k
GNO No Graphic

```

0-terminierter String <Version> muss als 3 Hexadezimalwerte interpretiert werden.
Z.B. 1144 1.1.68

Beispiel

```
char *strptr;
DWORD *dwptr;
unsigned char infostr[100];

if ( LslGetPLCInfoH(0, infostr, sizeof(infostr) ) == FALSE )
printf( "Error in LslGetPLCInfo\n" );
else
{
infostr[sizeof(infostr)-1] = '\0';
dwptr = (DWORD *)infostr;
printf("LslGetPLCInfo: usrCodeStart = 0x%04X\n", *dwptr++ );
printf("LslGetPLCInfo: usrCodeLength = 0x%04X\n", *dwptr++ );
printf("LslGetPLCInfo: usrDataStart = 0x%04X\n", *dwptr++ );
printf("LslGetPLCInfo: usrDataLength = 0x%04X\n", *dwptr++ );
printf("LslGetPLCInfo: Ofs EXE-CodeStart = 0x%04X\n", *dwptr++ );

strptr = (char *)dwptr;
printf("LslGetPLCInfo: BIOS.PLNAME = %s\n", (char *)strptr);
strptr += (strlen(strptr) + 1);
printf("LslGetPLCInfo: BIOS.GRAPHMODE = %s\n", (char *)strptr);
strptr += (strlen(strptr) + 1);
printf("LslGetPLCInfo: Version = %s\n", (char *)strptr);
strptr += (strlen(strptr) + 1);
printf("LslGetPLCInfo: printf("LslGetPLCInfo: Bootimage Type = %s\n", (char *)&strptr[2]);
strptr += (strlen(strptr) + 1);
if (strcmp((char *)strptr, "LSE") == 0)
printf("LslGetPLCInfo: printf("LslGetPLCInfo: LSE Kernel supported");
else
printf("LslGetPLCInfo: printf("LslGetPLCInfo: LSE Kernel not supported");
}
```

6.1.9.18 LslGetProjectInfo

Entspricht der Funktion [LslGetProjectInfoH\(\)](#) mit ocbNum = 0.

```
external "C" LSL_BOOL LASAL32_EXPORTS LslGetProjectInfo(void *pRetData0);
```

6.1.9.19 LslGetProjectInfoEx

Entspricht LslGetProjectInfoExH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetProjectInfoEx(
    PrjHeader* pRetData0,
    uint32_t bufferSize,
    uint32_t* nRead
);
```

6.1.9.20 LslGetProjectInfoExH

Die Funktion liefert einige Informationen über das Projekt. Es werden nur jene Module berücksichtigt, die mit der Voreinstellung von modTypeMask UND-verknüpft den voreingestellten Wert von modTypeEq ergeben. Die Voreinstellungen werden mit den Funktionen LslSetDfltModTypeParams() bzw. LslSetDfltModTypeParamsH() gesetzt.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetProjectInfoExH(
    int32_t          ocbNum,
    PrjHeader*       pRetData0,
    uint32_t         bufferSize,
    uint32_t*        nRead
);
```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
pRetData0	PrjHeader*	Zeiger auf den Ergebnispuffer (Aufbau des Ergebnispuffers siehe weiter unten)	
bufferSize	uint32_t	Größe des Ergebnispuffers pRetData0 in Byte. Der Puffer muss mindestens so groß sein, wie eine PrjHeader-Struktur. Er sollte aber 8 Byte größer sein, damit ein maximal langer Projektname nicht für error_master_count und ModTypes gekürzt werden muss.	
nRead	uint32_t*	Zeiger auf eine Variable, wo die Anzahl der Datenbytes im Ergebnispuffer eingetragen wird. Dieser Parameter darf NULL sein, wenn die Information nicht benötigt wird.	
Rückgabeparameter	Typ	Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	

Aufbau des Ergebnispuffers: zuerst eine PrjHeader-Struktur. Dahinter uint32_t error_master_count und je nach OS-Version uint32_t ModTypes oder 0 (siehe unten).



error_master_count und ModTypes werden direkt hinter den Projektamen kopiert und sind daher u.U. nicht long-aligned.

Byte 0,1,2,3	Prüfsumme des Projekts (Applikation + Loader)
Byte 4,5,6,7	Anzahl der aktiven Module

Byte 8	Anzahl der anstehenden Messages
Byte 9,10,11	Dummy1, Dummy2, Dummy3
Byte 12-n	Projektname (0-terminiert)
Byte n+1,2,3,4	error_master_count (alle aufgetretenen (Error-)Messages)
Byte n+5,6,7,8	LasalOS Version < 1.1.59: 0 LasalOS Version >= 1.1.59: ModTypes (ODER-Verknüpfung der berücksichtigten Modultypen)

PrjHeader-Struktur

Offset	Type	Description
0	unsigned long	Prüfsumme des Projekts (Applikation + Loader).
4	unsigned long	Anzahl der aktiven Module.
8	char	MsgCounter
9	char	Dummy1
10	char	Dummy2
11	char	Dummy3
12	char [64]	Projekt Name (0-terminiert).

Beispiel

```

char PrjHeaderBuffer[sizeof(PrjHeader)+8];
PrjHeader* pPrjHeader = (PrjHeader*)PrjHeaderBuffer;
uint32_t nRead = 0;
uint32_t* pData = NULL;

if (LslGetProjectInfoEx(pPrjHeader, sizeof(PrjHeaderBuffer), &nRead))
{
    printf("Checksum: 0x%08X\n", pPrjHeader->Prjchk);
    printf("Anzahl Module: %d\n", pPrjHeader->ModEnabled);
    printf("Message Counter: %d\n", pPrjHeader->Status.MsgCounter);
    printf("Project Name: %s\n\n", pPrjHeader->PrjName);
    printf("nRead: %d\n", nRead);

    pData = (uint32_t*)((char*)&pPrjHeader->PrjName + strlen(pPrjHeader->PrjName)+1);
    printf("ErrorCounter: %d\n", *pData);
    printf("ModTypes: 0x%08x\n", *(pData+1));
}
else
    printf("LslGetProjectInfoEx() failed(0x%08X)\n", GetLastError());

```

6.1.9.21 LslGetProjectInfoH

Die Funktion LslGetProjectInfoH() ruft Projektinformationen auf.

```
external "C" LSL_BOOL LASAL32_EXPORTS LslGetProjectInfoH(
    int32_t      ocbNum,
    PrjHeader    pRetData0
);
```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
pRetData0	PrjHeader	Zeiger auf den Ergebnispuffer. Aufbau des Ergebnispuffers siehe unter LslGetProjectInfoExH().	
		Offset	Typ
		0	Unsigned long
		4	Unsigned long
		8	Unsigned long
		12	Char [64]
		76	Char [180]
Rückgabeparameter		Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	

Beispiel

```
struct PrjHeader {
    unsigned long Prjchk;
    unsigned long ModEnabled;
    unsigned long Res1;
    char PrjName[64];
    char Res2[180];
} Ph;

if ( LslGetProjectInfo( (void *)&Ph ) == FALSE )
    printf( "Error in LslGetProjectInfo\n" );
else
{
    printf( "LslGetProjectInfo: Check CheckSUM      : 0x%08lx\n",
            Ph.Prjchk );
```

```

printf( "LslGetProjectInfo: Modules enabled      : 0x%08lx\n",
       Ph.ModEnabled );
printf( "LslGetProjectInfo: Project name          : %s\n",
       Ph.PrjName );
}

```

6.1.9.22 LslReadDateTime

Entspricht LslReadDateTimeH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslReadDateTime(
LSL_SYSTEMTIME *pSysTime
);
```

6.1.9.23 LslReadDateTimeH

Die Funktion liefert in pSysTime Datum und Uhrzeit der SPS zurück.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslReadDateTimeH(
    int32_t      ocbNum,
    LSL_SYSTEMTIME *pSysTime
);
```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
pSysTime	LSL_SYSTEMTIME*	Zeiger auf eine LSL_SYSTEMTIME-Struktur, wo die Werte von der SPS eingetragen werden				
Rückgabeparameter	Typ	Beschreibung				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 25%; padding: 2px;">TRUE</td><td style="width: 75%; padding: 2px;">Funktion erfolgreich</td></tr> <tr> <td style="padding: 2px;">FALSE</td><td style="padding: 2px;">Fehler</td></tr> </table> <p style="margin-top: 10px;">Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>	TRUE	Funktion erfolgreich	FALSE	Fehler
TRUE	Funktion erfolgreich					
FALSE	Fehler					

Beispiel

```

LSL_SYSTEMTIME SysTime;

if (LslReadDateTime(&SysTime))
    printf("%d.%d.%d\n", SysTime.wDay, SysTime.wMonth, SysTime.wYear);
else
    printf("LslReadDateTime() failed(0x%08X)\n", GetLastError());

```

6.1.9.24 LslSetDateTime

Entspricht LslSetDateTimeH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslSetDateTime(
LSL_SYSTEMTIME *pSysTime
);
```

6.1.9.25 LslSetDateTimeH

Die Funktion setzt Datum und Uhrzeit der SPS auf die in pSysTime angegebenen Werte.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslSetDateTimeH(
    int32_t      ocbNum,
    LSL_SYSTEMTIME *pSysTime
);
```

Übergabeparameter	Typ	Beschreibung					
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())					
pSysTime	LSL_SYSTEMTIME*	Zeiger auf eine LSL_SYSTEMTIME-Struktur mit Datum und Uhrzeit für die SPS					
Übergabeparameter	Typ	Beschreibung					
		<table border="1"> <tr> <td>TRUE</td><td>Funktion erfolgreich</td></tr> <tr> <td>FALSE</td><td>Fehler</td></tr> </table>		TRUE	Funktion erfolgreich	FALSE	Fehler
TRUE	Funktion erfolgreich						
FALSE	Fehler						
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.					

Beispiel

```
LSL_SYSTEMTIME SysTime;

SysTime.wMinute = 0;
if (LslSetDateTime(&SysTime))
    printf("LslSetDateTime() OK\n");
else
    printf("LslSetDateTime() failed(0x%08X)\n", GetLastError());
```

6.1.9.26 LslSyssernumCommand

Entspricht LslSyssernumCommandH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslSyssernumCommand(
    int32_t      iCmd,
    void        *pInput,
    int32_t      inLen,
```

```
void      *pOutput,
int32_t   outLen
);
```

6.1.9.27 LslSyssernumCommandH

Diese Funktion liefert die Seriennummer der SPS oder die Seriennummer des angegebenen Laufwerks als 0-terminierten ASCII-String.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslSyssernumCommandH(
    int32_t   ocbNum,
    int32_t   iCmd,
    void      *pInput,
    int32_t   inLen,
    void      *pOutput,
    int32_t   outLen
);
```

Übergabeparameter		Typ	Beschreibung				
ocbNum		int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
iCmd		int32_t	1 Seriennummer der SPS lesen				
			2 Seriennummer des in pInput angegebenen Laufwerks lesen				
pInput		void*	Zeiger auf den Laufwerksbuchstaben. Der Parameter hat keine Bedeutung, wenn iCmd nicht 2 ist.				
inLen		int32_t	Länge der Daten in pInput in Byte				
pOutput		void*	Zeiger auf einen Puffer, wo die Seriennummer als 0-terminierter ASCII-String eingetragen wird				
outLen		int32_t	Länge des Puffer pOutput in Byte. Der Puffer muss mindestens 20 Byte lang sein.				
Rückgabeparameter		Typ	Beschreibung				
			<table border="1"> <tr> <td>TRUE</td><td>Funktion erfolgreich</td></tr> <tr> <td>FALSE</td><td>Fehler</td></tr> </table>	TRUE	Funktion erfolgreich	FALSE	Fehler
TRUE	Funktion erfolgreich						
FALSE	Fehler						
			Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.				

Beispiel

```
char SerNum[32];
```

```

#define SERNUM_PLC      1
#define SERNUM_DRIVE    2

if (LslSyssernumCommand(SERNUM_PLC, NULL, 0, SerNum, sizeof(SerNum)))
    printf("Serial number PLC: %s\n", SerNum);
else
    printf("LslSyssernumCommand() failed(0x%08X)\n", GetLastError());

if (LslSyssernumCommand(SERNUM_DRIVE, "C", 1, SerNum, sizeof(SerNum)))
    printf("Serial number Drive C: %s\n", SerNum);
else
    printf("LslSyssernumCommand() failed(0x%08X)\n", GetLastError());

```

6.1.9.28 SetCommand

Entspricht der Funktion [SetCommandH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS SetCommand(
    uint8_t  uCommand
);

```

6.1.9.29 SetCommandH

Die Funktion [SetCommandH\(\)](#) sendet einen Befehl an die SPS.

```

extern "C" LSL_BOOL LASAL32_EXPORTS SetCommandH(
    int32_t  ocbNum,
    uint8_t  uCommand
);

```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
uCommand	uint8_t	Dieser Parameter gibt den Typ des Befehls an. Zulässig sind nur folgende Befehle: <ul style="list-style-type: none"> Reset (3) Run (4) 	
Rückgabeparameter	Typ	Beschreibung	
	LSL_BOOL	TRUE Funktion erfolgreich FALSE Fehler	
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	

Beispiel

```
if ( SetCommandH(0, 3 ) == TRUE )  
    printf( "SetCommand( 3 ) O.K. (RESET)\n" );  
else  
    printf( "Error in SetCommand( 3 ) (RESET)\n" );
```

6.1.10 Administrativfunktionen

6.1.10.1 SetMultiThreadSupport

Die Funktion SetMultiThreadSupport() kann verwendet werden, um die Multithread-Unterstützung für die Online-Funktionen zu aktivieren oder zu deaktivieren. Wenn Multithread-Unterstützung aktiviert ist, werden alle Online-Funktionen mit einem kritischen Abschnitt-Objekt geschützt.

In Single-Thread-Programmen kann die Multithread-Unterstützung deaktiviert werden. Dadurch verringert sich der Aufwand zum Schutz der Online-Funktionen. Bitte achten sie darauf, dass die Onlinefunktionen nicht Thread-sicher sind, wenn die Multithread-Unterstützung deaktiviert ist.

Die Multithread-Unterstützung ist standardmäßig aktiviert.

```
extern "C" void WINAPI SetMultiThreadSupport(unsigned long val);
```

Übergabeparameter	Typ	Beschreibung
val		<p>Gibt an, ob die Multithread-Unterstützung deaktiviert oder aktiviert werden soll.</p> <p>0 Deaktivieren 1 Aktivieren</p>

Es ist wichtig zu beachten, dass in Lasal32.dll-Versionen vor 1.34 die Anwendung für den Schutz der Funktionen vor dem Neueintritt in Multi-Thread-Anwendungen zuständig ist.

Anforderungen

Version Lasal32.dll ab 1.34 benötigt.

6.1.11 Dateifunktionen

6.1.11.1 CB_FORMAT_FUNCTYPE

Callback-Funktion von [LslFormatDrive\(\)](#).

```
typedef long __cdecl CB_FORMAT_FUNCTYPE(
    LSL32_FORMAT_PROGRESS_INFO *pFormatProgressInfo);
```

Übergabeparameter	Typ	Beschreibung
PFormatProgressInfo		Zeiger auf einen LSL32_FORMAT_PROGRESS_INFO Typ, der die Fortschrittsinformationen enthält.
Rückgabeparameter	Typ	Beschreibung
		Callback-Funktion wird durch die DLL aufgerufen, solange der Rückgabewert größer oder gleich 0 ist.

Information Struktur

```
typedef struct
{
    char DeviceName[ 32 ];

#define LSL32_FORMAT_PROGRESS_READY      0
#define LSL32_FORMAT_PROGRESS_BUSY       1
#define LSL32_FORMAT_PROGRESS_DONE      2
#define LSL32_FORMAT_PROGRESS_ERROR     3

    DWORD Total;
    DWORD Completed;
    DWORD Status;

} LSL32_FORMAT_PROGRESS_INFO;
```

DeviceName	CHAR	Gerätename
Gesamt	DWORD	Die Anzahl der Sektoren, die verarbeitet werden müssen
Abgeschlossen	DWORD	Die Anzahl der Sektoren, die erfolgreich verarbeitet wurden
Zustand	DWORD	Statusinformationen über den Formatverlauf. <ul style="list-style-type: none"> • LSL32_FORMAT_PROGRESS_READY Das Laufwerk ist zum Formatieren bereit (intern verwendet). • LSL32_FORMAT_PROGRESS_BUSY Die Formatierung läuft. • LSL32_FORMAT_PROGRESS_DONE Formatierung erfolgreich abgeschlossen.

- | | | |
|--|--|----------------------------------------------------------------------------------------------------------------------|
| | | <ul style="list-style-type: none"> • LSL32_FORMAT_PROGRESS_ERROR
Formatierung fehlerhaft beendet. |
|--|--|----------------------------------------------------------------------------------------------------------------------|

Anforderungen

Lasal32.dll	ab 01.01.018
LasalOS	ab 01.01.067

Beispiel

```

LSL_BOOL rc;
int iRC;

rc = LslFormatDrive("E:\\\\", 0, &iRC, FormatProgressCallback);

if (rc == FALSE)
Error
else
{
    ; // command successful
    if (iRC < 0)
        ; // error of the format operation
    else
        ; // format successful finished
}

long FormatProgressCallback(LSL32_FORMAT_PROGRESS_INFO * pProgressInfo)
{
    printf("DeviceName: %s, Total: %d, Completed: %d, Status: %d\\n",
        pProgressInfo->DeviceName,
        pProgressInfo->Total,
        pProgressInfo->Completed,
        pProgressInfo->Status);

    return (0); // the function will be called again ( >= 0)

// return (-1); // the function won't be called again ( < 0)
}

```

6.1.11.2 FileInfo

Entspricht der Funktion [FileInfoH\(\)](#) mit ocbNum = 0.

```

LSL_BOOL LASAL32_EXPORTS FileInfo(
    char          *RBuf,
    const char    *Dest,
    uint32_t      BufLen
);

```

6.1.11.3 FileInfoH

Die Funktion FileInfoH() liefert Informationen über eine Datei in der SPS zurück.

```
extern "C" LSL_BOOL LASAL32_EXPORTS FileInfo(
    int32_t          ocbNum,
    char             *RXbuf,
    char             *dest,
    unsigned long    buflen
);
```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen()).				
RXbuf	CHAR	<p>Zeiger auf den Puffer, der die Informationen erhält. Dieser Zeiger muss auf die folgende Struktur zeigen:</p> <pre>typedef struct LSL_FILE_INFO { char FileName[8]; char Extension[3]; uint8_t cAttributes; _DDE_DATIM dtDateTime; // of last modification uint32_t lFileSize; } LSL_FILE_INFO;</pre> <p>Das Element dtDateTime ist mit den folgenden Informationen gefüllt:</p> <pre>typedef struct _DDE_DATIM { uint32_t Second2:5; // seconds divided by 2 (0..30) uint32_t Minute:6; // 0..59 uint32_t Hour:5; // 0..23 uint32_t Day:5; // 1..31 uint32_t Month:4; // 1..12 uint32_t Year1980:7; // Years since 1980 } _DDE_DATIM;</pre>				
dest	CHAR	Zeiger auf einen 0-terminierten String, der den Namen der Datei in der SPS angibt. Es sollte immer ein absoluter Pfadname spezifiziert werden, der aus dem Laufwerk, dem Verzeichnis und dem Dateinamen bestehen.				
buflen	UNSIGNED LONG	Gibt die Länge des Puffers an				
Rückgabeparameter	Typ	Beschreibung				
		<table border="1"> <tr> <td>TRUE</td><td>Funktion erfolgreich</td></tr> <tr> <td>FALSE</td><td>Fehler</td></tr> </table> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>	TRUE	Funktion erfolgreich	FALSE	Fehler
TRUE	Funktion erfolgreich					
FALSE	Fehler					

Beispiel

```

LSL_FILE_INFO fileInfo;
if (FileInfo((char*)fileInfo, "C:\\\\testdir", sizeof(fileInfo)))
{
    //File name and extension
    printf("Attributes of %.8s.%3s: ", fileInfo.FileName, fileInfo.Extension);
    //Date and time
    year = fileInfo.dtDateTime.Year1980 + 1980;
    month = fileInfo.dtDateTime.Month;
    day = fileInfo.dtDateTime.Day;
    hour = fileInfo.dtDateTime.Hour;
    minute = fileInfo.dtDateTime.Minute;
    second2 = fileInfo.dtDateTime.Second2;
    printf("%d.%02d.%02d ", year, month, day);
    printf("%02d:%02d:%02d\\n", hour, minute, second2);
    // Attributes
    printf("%c", (fileInfo.cAttributes & LSL32_ATTR_DIR) ? 'd' : '-');
    printf("%c", (fileInfo.cAttributes & LSL32_ATTR_READ_ONLY) ? 'r' : '-');
    printf("%c", (fileInfo.cAttributes & LSL32_ATTR_HIDDEN) ? 'h' : '-');
    printf("%c", (fileInfo.cAttributes & LSL32_ATTR_SYSTEM) ? 's' : '-');
    printf("%c", (fileInfo.cAttributes & LSL32_ATTR_ARCHIVE) ? 'a' : '-');
    printf("\\n");
    //File size
    printf("Size: %d\\n", fileInfo.lFileSize);
}

```

6.1.11.4 FileLoad

Entspricht der Funktion [FileLoadH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS FileLoad(
    char          *Rdbuf,
    const char    *dest,
    uint32_t      datalen,
    CB_FUNCTYPE   *callback,
    int32_t       offs
);

```

6.1.11.5 FileLoadH

Die Funktion FileLoadH() liest die Daten aus einer Datei auf der SPS, ab der Position, die durch den Parameter offs festgelegt ist.

```

typedef unsigned long __cdecl CB_FUNCTYPE(unsigned long);

extern "C" LSL_BOOL LASAL32_EXPORTS FileLoadH(
    int32_t       ocbNum,
    char          *Rdbuf,
    char          *dest,
    unsigned long  datalen,
    CB_FUNCTYPE   *callback,

```

```
long          offs
);
```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen()).	
RXbuf	CHAR	Zeiger auf den Puffer, der die Daten aus der Datei auf der SPS überträgt	
dest	CHAR	Zeiger auf einen 0-terminierten String, der den Namen der Datei in der SPS angibt. Es sollte immer ein absoluter Pfadname spezifiziert werden, der aus dem Laufwerk, dem Verzeichnis und dem Dateinamen bestehen.	
dateien	UNSIGNED LONG	Gibt die Anzahl der aus der Datei zu lesenden Byte an. Wenn die angegebene datalen die Länge der zu lesenden Datei überschreitet, dann wird die Funktion nicht ausgeführt.	
callback	CB_FUNCTYPE*	Zeiger auf eine CB_FUNCTYPE Callback-Funktion. Bei der Übergabe der Daten aus der SPS ist es möglich, dass die Daten in Blöcke unterteilt werden. Die Callback-Funktion wird vor der Beantragung des Blocks mit der Anzahl der verbleibenden Blöcke als Parameter aufgerufen. Die Callback-Funktion kann verwendet werden, um z. B. einen Fortschrittsbalken zu aktualisieren. Wenn der Parameter callback NULL ist, wird keine Callback-Funktion aufgerufen.	
offs	LONG	Gibt die 0-Position in der Datei an, von wo das Lesen der Daten beginnen soll	
Übergabeparameter	Typ	Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	

[FileInfo\(\)](#) oder [FileInfoH\(\)](#) kann verwendet werden, um die Länge der Datei in der SPS abzufragen.

Beispiel

```
static BOOL s_StdCallbackInit = FALSE;
static void StdCallback_Init()
{
    s_StdCallbackInit = FALSE;
}

static uint32_t __cdecl StdCallback(uint32_t val)
{
```

```
if (s_StdCallbackInit == FALSE)
{
    s_StdCallbackInit = TRUE;
    printf("Callback Init : %u      \n", val);
}
else
    printf("Callback Val  : %u      \r", val);
return CBRETURN_CONTINUE;
}

if (::Online("10.10.170.190", 0, 0, 0, 0))
{
    LSL_FILE_INFO fileInfo;
    if (FileInfo((char *)&fileInfo, "C:\\\\testfile.txt", sizeof(fileInfo)))
    {
        void* pBuff = malloc(fileInfo.lFileSize);

        printf("Reading %u Bytes\n", fileInfo.lFileSize);

        StdCallback_Init();
        if (::FileLoad((char*)pBuff, "C:\\\\testfile.txt", fileInfo.lFileSize,
                      StdCallback, 0))
        {
            //ToDo: Daten verarbeiten
        }
        else
        {
            printf("FileLoad failed 0x%08X\n", GetLastError());
        }
        free(pBuff);
    }
    else
        printf("FileInfo failed 0x%08X\n", GetLastError());
    ::Offline();
}
else
    printf("Online failed 0x%08X\n", GetLastError());
```

6.1.11.6 FileSave

Entspricht der Funktion [FileSaveH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS FileSave(
    const char      *dest,
    char            *pData,
    uint32_t        len,
    CB_FUNCTYPE    *callback,
    uint32_t        attrib
);
```

6.1.11.7 FileSaveEx

Entspricht der Funktion [FileSaveExH\(\)](#) mit ocbNum = 0.

```
typedef unsigned long __cdecl CB_FUNCTYPE(unsigned long);

extern "C" LSL_BOOL LASAL32_EXPORTS FileSaveEx(
    const char      *dest,
    const char      *pData,
    uint32_t        len,
    CB_FUNCTYPE    *callback,
    int32_t         *errCodeRemote
);
```

6.1.11.8 FileSaveExH

Die Funktion FileSaveExH() überträgt die Daten in eine Datei auf der SPS. Wenn die angegebene Datei schon vorhanden ist und das Attribut Read-Only nicht gesetzt ist, dann wird die Datei überschrieben. Diese Funktion meldet einen SPS-Fehler-Code im Fall eines Remote-Errors.

```
typedef unsigned long __cdecl CB_FUNCTYPE(unsigned long);

extern "C" LSL_BOOL LASAL32_EXPORTS FileSaveExH(
    int32_t         ocbNum,
    const char      *dest,
    const char      *pData,
    uint32_t        len,
    CB_FUNCTYPE    *callback,
    int32_t         *errCodeRemote
);
```

Transfer parameters	Type	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
dest	CONST CHAR	Zeiger auf einen 0-terminierten String, der den Namen der Datei in der SPS angibt. Es sollte immer ein absoluter Pfadname spezifiziert werden, der aus dem Laufwerk, dem Verzeichnis und dem Dateinamen bestehen.
pData	CONST CHAR	Zeiger auf den Puffer mit den Daten, die in die Datei geschrieben werden sollen
len	uint32_t	Legt die Anzahl von Byte fest, die in die Datei geschrieben werden sollen
callback	CB_FUNCTYPE*	Zeiger auf eine CB_FUNCTYPE Callback-Funktion. Bei der Übergabe der Daten an die SPS ist es möglich, dass die Daten in Blöcke unterteilt werden. Die Callback-Funktion wird vor der Übertragung eines Blocks mit der Anzahl der verbleibenden Blöcke als Parameter aufgerufen. Die Callback-Funktion kann verwendet werden, um die Übertragung zu unterbrechen.

		werden, um z. B. einen Fortschrittsbalken zu aktualisieren. Wenn der Parameter callback NULL ist, wird keine Callback-Funktion aufgerufen.																											
errCodeRemote	int32_t	<p>Zeiger auf eine Variable, der den Fehler-Code von der abgesetzten SPS erhält. Dieser Parameter ist nur dann sinnvoll, wenn die Funktion fehlerhaft ist und GetLastError() ERROR_SIGMA32_REMOTE_ERROR zurückliefert.</p> <p>Zu beachten ist, dass die Betriebssystemversionen älter als 5.43 keinen Fehler-Code melden. In diesem Fall setzt die FileSaveExH-Funktion den Wert von errCodeRemote auf LSLOS_ERROR_GENERAL. Dies ist ein unbestimmter Fehler-Code. Für eine Liste der möglichen LasalOS-Fehler-Codes sehen Sie bitte in der Dokumentation LasalOS nach.</p> <p>Die folgende Tabelle enthält eine Liste der LasalOS-Fehler-Codes. Zu beachten ist, dass diese Fehler-Codes nur ein Auszug aller möglichen Fehler-Codes sind. Für eine Liste der anderen möglichen Fehler-Codes sehen Sie bitte in der Dokumentation LasalOS nach.</p> <table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>0xA00080 13</td> <td>General error, not categorized.</td> <td>LSLOS_ERROR_GENERAL</td> </tr> <tr> <td>0xA00100 00</td> <td>Allgemeinen Dateifehler, nicht kategorisiert</td> <td>LSLOS_FILE_ERROR_GENERAL</td> </tr> <tr> <td>0xA00180 03</td> <td>Die Dateinamen, Verzeichnisnamen oder Volume Label Syntax sind nicht korrekt.</td> <td>LSLOS_ERROR_INVALID_FILENAME</td> </tr> <tr> <td>0xA00180 04</td> <td>Das System kann das angegebene Laufwerk nicht finden.</td> <td>LSLOS_ERROR_DRIVE_NOT_FOUND</td> </tr> <tr> <td>0xA00180 09</td> <td>Das System kann die angegebene Datei nicht finden.</td> <td>LSLOS_ERROR_FILE_NOT_FOUND</td> </tr> <tr> <td>0xA00180 10</td> <td>Zugriff wird verweigert.</td> <td>LSLOS_ERROR_ACCESS_DENIED</td> </tr> <tr> <td>0xA00180 13</td> <td>Das System kann den angegebenen Pfad nicht finden.</td> <td>LSLOS_ERROR_PATH_NOT_FOUND</td> </tr> <tr> <td>0xA00180 16</td> <td>Es ist nicht ausreichend Platz auf der Festplatte.</td> <td>LSLOS_ERROR_DISK_FULL</td> </tr> </tbody> </table>	Code	Description	Name	0xA00080 13	General error, not categorized.	LSLOS_ERROR_GENERAL	0xA00100 00	Allgemeinen Dateifehler, nicht kategorisiert	LSLOS_FILE_ERROR_GENERAL	0xA00180 03	Die Dateinamen, Verzeichnisnamen oder Volume Label Syntax sind nicht korrekt.	LSLOS_ERROR_INVALID_FILENAME	0xA00180 04	Das System kann das angegebene Laufwerk nicht finden.	LSLOS_ERROR_DRIVE_NOT_FOUND	0xA00180 09	Das System kann die angegebene Datei nicht finden.	LSLOS_ERROR_FILE_NOT_FOUND	0xA00180 10	Zugriff wird verweigert.	LSLOS_ERROR_ACCESS_DENIED	0xA00180 13	Das System kann den angegebenen Pfad nicht finden.	LSLOS_ERROR_PATH_NOT_FOUND	0xA00180 16	Es ist nicht ausreichend Platz auf der Festplatte.	LSLOS_ERROR_DISK_FULL
Code	Description	Name																											
0xA00080 13	General error, not categorized.	LSLOS_ERROR_GENERAL																											
0xA00100 00	Allgemeinen Dateifehler, nicht kategorisiert	LSLOS_FILE_ERROR_GENERAL																											
0xA00180 03	Die Dateinamen, Verzeichnisnamen oder Volume Label Syntax sind nicht korrekt.	LSLOS_ERROR_INVALID_FILENAME																											
0xA00180 04	Das System kann das angegebene Laufwerk nicht finden.	LSLOS_ERROR_DRIVE_NOT_FOUND																											
0xA00180 09	Das System kann die angegebene Datei nicht finden.	LSLOS_ERROR_FILE_NOT_FOUND																											
0xA00180 10	Zugriff wird verweigert.	LSLOS_ERROR_ACCESS_DENIED																											
0xA00180 13	Das System kann den angegebenen Pfad nicht finden.	LSLOS_ERROR_PATH_NOT_FOUND																											
0xA00180 16	Es ist nicht ausreichend Platz auf der Festplatte.	LSLOS_ERROR_DISK_FULL																											

		0xA00180 1A	Die Medien im Laufwerk haben sich möglicherweise geändert.	LSLOS_ERROR_MEDIA_CHANGED
		0xA00180 1D	Das Gerät ist nicht bereit.	LSLOS_ERROR_NOT_READY
		0xA00180 1E	Der Datenträger ist schreibgeschützt.	LSLOS_ERROR_WRITE_PROTECT
Rückgabeparameter	Typ	Beschreibung		
		TRUE	Funktion erfolgreich	
		FALSE	Fehler	
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.				

Anforderungen

Lasal32.dll ab 1.36

Beispiel

```
static BOOL s_StdCallbackInit = FALSE;
static void StdCallback_Init()
{
    s_StdCallbackInit = FALSE;
}

static uint32_t __cdecl StdCallback(uint32_t val)
{
    if (s_StdCallbackInit == FALSE)
    {
        s_StdCallbackInit = TRUE;
        printf("Callback Init : %u      \n", val);
    }
    else
        printf("Callback Val   : %u      \r", val);
    return CBRETURN_CONTINUE;
}

if (::Online("10.10.170.190", 0, 0, 0, 0))
{
    LSL_FILE_INFO fileInfo;
    int32_t errCodeRemote;
    uint32_t error;
```

```

if (FileInfo((char *)&fileInfo, "C:\\testfile.txt", sizeof(fileInfo)))
{
    void* pBuff = malloc(fileInfo.lFileSize);

    printf("Reading %u Bytes\n", fileInfo.lFileSize);

    StdCallback_Init();
    if (::FileLoad((char*)pBuff, "C:\\testfile.txt", fileInfo.lFileSize,
        StdCallback, 0))
    {
        StdCallback_Init();
        if (!::FileSaveEx((char*)"C:\\backup.txt", pBuff, fileInfo.lFileSize,
            StdCallback, &errCodeRemote))
        {
            error = GetLastError();
            printf("FileSaveEx failed 0x%08X\n", error);
            if (error == ERROR_SIGMA32_REMOTE_ERROR)
                printf("PLC reported error: 0x%08X\n", error);
        }
    }
    else
    {
        printf("FileLoad failed 0x%08X\n", GetLastError());
    }
    free(pBuff);
}
else
    printf("FileInfo failed 0x%08X\n", GetLastError());

::Offline();
}
else
    printf("Online failed 0x%08X\n", GetLastError());

```

6.1.11.9 FileSaveH

Die Funktion `FileSaveH()` überträgt die Daten in eine Datei auf der SPS. Wenn die angegebene Datei vorhanden ist, werden ihre Inhalte zerstört.

Im Fall eines Remote-Fehlers stellt diese Funktion keinen Fehler-Code aus der SPS zur Verfügung. Verwenden Sie die Funktion [FileSaveExH\(\)](#), wenn Sie am SPS-Fehler-Code interessiert sind.

```

typedef unsigned long _cdecl CB_FUNCTYPE(unsigned long);

extern "C" LSL_BOOL LASAL32_EXPORTS FileSaveH(
    int32_t          ocbNum,
    const char       *dest,
    char             *pData,
    uint32_t          len,
    CB_FUNCTYPE      *callback,
    uint32_t          attrib
);

```

Übergabeparameter	Typ	Beschreibung											
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())											
dest	CONST CHAR	Zeiger auf einen 0-terminierten String, der den Namen der Datei in der SPS angibt. Es sollte immer ein absoluter Pfadname spezifiziert werden, der aus dem Laufwerk, dem Verzeichnis und dem Dateinamen bestehen.											
pData	CHAR	Zeiger auf den Puffer mit den Daten, die in die Datei geschrieben werden sollen											
len	uint32_t	Legt die Anzahl von Bytes fest, die in die Datei geschrieben werden sollen											
callback	CB_FUNCTYPE*	Zeiger auf eine CB_FUNCTYPE Callback-Funktion. Bei der Übergabe der Daten an die SPS ist es möglich, dass die Daten in Blöcke unterteilt werden. Die Callback-Funktion wird vor der Übertragung eines Blocks mit der Anzahl der verbleibenden Blöcke als Parameter aufgerufen. Die Callback-Funktion kann verwendet werden, um z. B. einen Fortschrittsbalken zu aktualisieren. Wenn der Parameter callback NULL ist, wird keine Callback-Funktion aufgerufen.											
attrib	uint32_t	<p>Gibt die Dateiattribute für die Datei an. Eine Kombination der folgenden Attribute ist möglich:</p> <table border="1"> <thead> <tr> <th>Attribut</th> <th>Bedeutung</th> </tr> </thead> <tbody> <tr> <td>LSL32_ATTR_READONLY (0x01)</td> <td>Die Datei kann nur gelesen werden.</td> </tr> <tr> <td>LSL32_ATTR_HIDDEN (0x02)</td> <td>Die Datei ist versteckt. Es ist nicht sichtbar in einem gewöhnlichen Verzeichnis.</td> </tr> <tr> <td>LSL32_ATTR_SYSTEM (0x04)</td> <td>Die Datei ist ein Teil vom Betriebssystem oder wird ausschließlich vom Betriebssystem verwendet.</td> </tr> <tr> <td>LSL32_ATTR_ARCHIVE (0x20)</td> <td>Die Datei sollte archiviert werden. Anwendungen benutzen dieses Attribut, um Dateien für die Sicherung oder Backups zu markieren.</td> </tr> </tbody> </table> <p>Dieser Parameter wird in Betriebssystemen ab Version 5.28 ignoriert!</p>		Attribut	Bedeutung	LSL32_ATTR_READONLY (0x01)	Die Datei kann nur gelesen werden.	LSL32_ATTR_HIDDEN (0x02)	Die Datei ist versteckt. Es ist nicht sichtbar in einem gewöhnlichen Verzeichnis.	LSL32_ATTR_SYSTEM (0x04)	Die Datei ist ein Teil vom Betriebssystem oder wird ausschließlich vom Betriebssystem verwendet.	LSL32_ATTR_ARCHIVE (0x20)	Die Datei sollte archiviert werden. Anwendungen benutzen dieses Attribut, um Dateien für die Sicherung oder Backups zu markieren.
Attribut	Bedeutung												
LSL32_ATTR_READONLY (0x01)	Die Datei kann nur gelesen werden.												
LSL32_ATTR_HIDDEN (0x02)	Die Datei ist versteckt. Es ist nicht sichtbar in einem gewöhnlichen Verzeichnis.												
LSL32_ATTR_SYSTEM (0x04)	Die Datei ist ein Teil vom Betriebssystem oder wird ausschließlich vom Betriebssystem verwendet.												
LSL32_ATTR_ARCHIVE (0x20)	Die Datei sollte archiviert werden. Anwendungen benutzen dieses Attribut, um Dateien für die Sicherung oder Backups zu markieren.												
Rückgabeparameter	Typ	Beschreibung											
		TRUE Funktion erfolgreich											

FALSE Fehler

Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion [GetLastError\(\)](#) auf.

Beispiel

```
static BOOL s_StdCallbackInit = FALSE;
static void StdCallback_Init()
{
    s_StdCallbackInit = FALSE;
}

static uint32_t __cdecl StdCallback(uint32_t val)
{
    if (s_StdCallbackInit == FALSE)
    {
        s_StdCallbackInit = TRUE;
        printf("Callback Init : %u      \n", val);
    }
    else
        printf("Callback Val  : %u      \r", val);
    return CBRETURN_CONTINUE;
}

if (::Online("10.10.170.190", 0, 0, 0, 0))
{
    LSL_FILE_INFO fileInfo;
    if (FileInfo((char *)&fileInfo, "C:\\testfile.txt", sizeof(fileInfo)))
    {
        void* pBuff = malloc(fileInfo.lFileSize);

        printf("Reading %u Bytes\n", fileInfo.lFileSize);

        StdCallback_Init();
        if (::FileLoad((char*)pBuff, "C:\\testfile.txt", fileInfo.lFileSize,
                      StdCallback, 0))
        {
            if (!::FileSave((char*)"C:\\backup.txt", pBuff, fileInfo.lFileSize,
                           StdCallback, LSL32_ATTR_READ_ONLY))
            {
                printf("FileSave failed 0x%08X\n", GetLastError());
            }
        }
        else
        {
            printf("FileLoad failed 0x%08X\n", GetLastError());
        }
        free(pBuff);
    }
    else
        printf("FileInfo failed 0x%08X\n", GetLastError());
}
```

```

    ::Offline();
}
else
    printf("Online failed 0x%08X\n", GetLastError());

```

6.1.11.10 FileSetAttributes

Entspricht FileSetAttributesH() mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS FileSetAttributes(
    const char* dest,
    uint32_t     attrib
);

```

6.1.11.11 FileSetAttributesH

Diese Funktion setzt die Datei-Attribute. Attribute, die in attrib nicht angegeben sind, werden gelöscht. FileInfoH() liefert die Datei-Attribute.

```

extern "C" LSL_BOOL LASAL32_EXPORTS FileSetAttributesH(
    int32_t      ocbNum,
    const char*  dest,
    uint32_t     attrib
);

```

Übergabeparameter	Type	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
dest	const char*	Zeiger auf einen 0-terminierten String, der den Namen der Datei in der SPS angibt. Es sollte immer ein absoluter Pfadname spezifiziert werden, der aus dem Laufwerk, dem Verzeichnis und dem Dateinamen bestehen.
attrib	uint32_t	<p>Gibt die Dateiattribute für die Datei an. Das Betriebssystem RTK unterstützt eine Kombination der folgenden Datei-Attribute:</p> <ul style="list-style-type: none"> • LSL32_ATTR_READ_ONLY • LSL32_ATTR_HIDDEN • LSL32_ATTR_SYSTEM • LSL32_ATTR_ARCHIVE <p>Alle anderen Betriebssysteme unterstützen nur das Datei-Attribut LSL32_ATTR_READ_ONLY.</p> <p>Das Datei-Attribut LSL32_ATTR_DIR kann bei allen Betriebssystemen nur gelesen werden.</p>
Rückgabeparameter	Typ	Beschreibung

		TRUE Funktion erfolgreich
		FALSE Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.		

Beispiel

```

LSL_FILE_INFO fileInfo;
if (FileInfo((char*)&fileInfo, "C:\\\\testfile.txt", sizeof(fileInfo)))
{
  printf("Attributes of C:\\\\testfile.txt: ");
  // Attributes
  printf("%c", (fileInfo.cAttributes & LSL32_ATTR_DIR) ? 'd' : '-');
  printf("%c", (fileInfo.cAttributes & LSL32_ATTR_READ_ONLY) ? 'r' : '-');
  printf("%c", (fileInfo.cAttributes & LSL32_ATTR_HIDDEN) ? 'h' : '-');
  printf("%c", (fileInfo.cAttributes & LSL32_ATTR_SYSTEM) ? 's' : '-');
  printf("%c", (fileInfo.cAttributes & LSL32_ATTR_ARCHIVE) ? 'a' : '-');
  printf("\n");

  fileInfo.cAttributes |= LSL32_ATTR_READ_ONLY; //RTK und Salamander u.a.
  fileInfo.cAttributes |= LSL32_ATTR_HIDDEN; //nur RTK
  fileInfo.cAttributes |= LSL32_ATTR_SYSTEM; //nur RTK
  fileInfo.cAttributes |= LSL32_ATTR_ARCHIVE; //nur RTK
  if (FileSetAttributes("C:\\\\testfile.txt", fileInfo.cAttributes) == FALSE)
    printf("FileSetAttributes() failed(0x%08X)\n", GetLastError());
}
else
  printf("FileInfo() failed(0x%08X)\n", GetLastError());

```

6.1.11.12 LsICheckDisk

Entspricht der Funktion [LsICheckDiskH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LsICheckDisk(
  const char          *DriveName,
  uint32_t             ulOption,
  uint32_t             ulFlags,
  CB_CHKDSK_FUNCTYPE  CheckDiskProgressInfo,
  uint32_t             *pErrors,
  int32_t              *pResult
);

```

Beispiel

```

rc = LsICheckDisk("C:\\\\",
                  LSL32_CHKDSK_OPTION_CORRECT,
                  0,
                  CheckDiskProgressInfoCB,
                  &ulErrors,

```

```

    &iRC);

long CheckDiskProgressInfoCB(char * strProgressInfo, unsigned long ulLen)
{
printf("%s\n", strProgressInfo);
return(0);
}

```

6.1.11.13 LslCheckDiskH

Funktion zur Überprüfung eines FAT-formatierten logischen Laufwerks. Dateisystemfehler können optional auch automatisch korrigiert werden.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslCheckDiskH(
    int32_t          ocbNum,
    const char       *DriveName,
    uint32_t          ulOption,
    uint32_t          ulFlags,
    CB_CHKDSK_FUNCTYPE CheckDiskProgressInfo,
    uint32_t          *pErrors,
    int32_t          *pResult
);

```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
DriveName	CONST CHAR	Zeiger auf den Laufwerksbuchstaben				
ulOption	uint32_t	Optionale Funktion 0 => Keine Korrektur, wenn Fehler im Dateisystem gefunden wurden. #define LSL32_CHKDSK_OPTION_CORRECT 0x01 => CheckDisk wird Dateisystemfehler korrigieren.				
ulFlags	uint32_t	Reserviert				
CheckDiskProgressInfo	CB_CHKDSK_FUNC_TYPE	Zeiger auf eine CB_CHKDSK_FUNC_TYPE Callback-Funktion, um Fortschrittsinformationen zu zeigen. Wenn keine Informationen erforderlich sind, kann diese Funktion Null sein.				
pErrors	uint32_t	Zeiger auf eine Variable, die die Anzahl der gefundenen Fehler enthält				
pResult	int32_t	Zeiger auf einen int Wert, der mit dem Operationsergebnis gefüllt wird <table border="1" style="margin-left: 20px;"> <tr> <td>≥0</td> <td>Erfolg</td> </tr> <tr> <td><0</td> <td>Negativer Fehler-Code</td> </tr> </table>	≥0	Erfolg	<0	Negativer Fehler-Code
≥0	Erfolg					
<0	Negativer Fehler-Code					
Rückgabeparameter	Typ	Beschreibung				

		TRUE Funktion erfolgreich
		FALSE Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.		

Callback Funktion

Bei Fehlern im Dateisystem wird CheckDisk rekursiv aufgerufen, bis alle Fehler behoben wurden, da die Korrektur eines Fehlers zu einem anderen Fehler führen kann. Dies wird durch die DLL ausgeführt und kann in der Callback-Funktion beobachtet werden.

```
typedef long __cdecl CB_CHKDSK_FUNCTYPE
    (char *strProgressInfo, unsigned long ProgressInfoLen);
```

Übergabeparameter	Typ	Beschreibung
strProgressInfo	CHAR	Zeiger auf den Elektrodenstrang, der die Informationen erhält
ProgressInfoLen	UNSIGNED LONG	Informationslänge
Rückgabeparameter	Typ	Beschreibung
		≥0 calcMaxSlavePath&tB; wird von der Applikation aufgerufen
		<0 Die Callback-Funktion wird nicht mehr aufgerufen.

Anforderungen

Lasal32.dll	ab 01.01.019
LasalOS	ab 01.01.079

6.1.11.14 LslFileDelete

Entspricht der Funktion [LslFileDeleteH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileDelete(
    const char    *ccFileDirectoryName,
    uint32_t      ulOption,
    int32_t       *pResult
);
```

6.1.11.15 LslFileDeleteH

Die Funktion LslFileDeleteH() löscht eine bestimmte Datei oder entfernt ein bestimmtes Verzeichnis.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileDeleteH(
    int32_t          ocbNum,
    const char      *ccFileDirectoryName,
    uint32_t         ulOption,
    int32_t          *pResult
);
```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
ccFileDirectoryName	CONST CHAR	Zeiger auf eine Datei oder einen Verzeichnisnamen	
ulOption	uint32_t	Datei löschen oder Verzeichnis entfernen. #define LSL32_DELETE_FILE 0 #define LSL32_REMOVE_DIR 1	
pResult	int32_t	Zeiger auf einen int Wert, der mit dem Operationsergebnis gefüllt wird.	
		≥0	Erfolg
		<0	Negativer Fehler-Code
Rückgabeparameter	Typ	Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.			

Anforderungen

Lasal32.dll	ab 01.01.018
LasalOS	ab 01.01.067

Beispiel

```
BOOL rc:  

int iRC;  

// delete a file
```

```

rc = LslFileDelete("C:\autoexec.lsl", LSL32_DELETE_FILE);
if (rc == false)
    ; // command not successful
else
    ; // command successful, iRC gives information of the operation
// remove a directory
rc = LslFileDelete("C:\TEMP", LSL32_REMOVE_DIR);
if (rc == false)
    ; // command not successful
else
    ; // command successful, iRC gives information of the operation

```

6.1.11.16 LslFindCloseH

Die Funktion `LslFindClose()` schließt vorangegangene Aufrufe von [LslFindFirst\(\)](#)/[LslFindNext\(\)](#).

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslFindCloseH(
    int32_t    ocbNum,
    int32_t    *pResult
);

```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
pResult	int32_t	<p>Zeiger auf einen INT Wert, der mit dem Operationsergebnis gefüllt wird.</p> <table border="1" style="margin-left: 20px;"> <tr> <td>≥0</td> <td>Erfolg</td> </tr> <tr> <td><0</td> <td>Negativer Fehler-Code</td> </tr> </table>	≥0	Erfolg	<0	Negativer Fehler-Code
≥0	Erfolg					
<0	Negativer Fehler-Code					
Rückgabeparameter	Typ	<p>Beschreibung</p> <table border="1" style="margin-left: 20px;"> <tr> <td>TRUE</td> <td>Funktion erfolgreich</td> </tr> <tr> <td>FALSE</td> <td>Fehler</td> </tr> </table> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>	TRUE	Funktion erfolgreich	FALSE	Fehler
TRUE	Funktion erfolgreich					
FALSE	Fehler					

6.1.11.17 LslFindFirstH

Die Funktion `LslFindFirstH()` sucht in einem Verzeichnis eine Datei, die bestimmten Kriterien entspricht.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslFindFirstH(
    int32_t    ocbNum,
    const char *namePattern,
    BYTE       attr,

```

```

BYTE      attrMask,
int       *pResult,
_DDE_INFO * fileInfo,
char      * fileName,
UINT      maxLength
);

```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
namePattern	CONST CHAR	Zeiger auf einen Dateinamen, der Wildcard-Zeichen '*' (Übereinstimmung mit Null oder mehreren Zeichen) enthalten kann bzw. "?" (genau dieses Zeichen abgleichen) und kann wahlweise einem Pfad vorangestellt werden. Wenn ein Pfad bereits besteht, darf es keine Wildcard-Zeichen enthalten. Aus Kompatibilität mit dem MS-DOS Muster "*.*" ist leise umgewandelt "**". Zur Suche nach Dateinamen die einen oder mehrere Punkte erfüllen sollen, verwenden Sie "?*.*" oder "*.*?".
attr	BYTE	Gibt einen Satz von allen Dateiattributen aus, die eine Datei benötigt um die Anfrage abzulegen. Jede Kombination der folgenden Flags kann für die Parameter Attr und AttrMask festgelegt werden: <ul style="list-style-type: none"> • LSL32_ATTR_READ_ONLY • LSL32_ATTR_HIDDEN • LSL32_ATTR_SYSTEM • LSL32_ATTR_DIR • LSL32_ATTR_ARCHIVE
attrMask	BYTE	Definiert den Satz von Attributen, die mit Attr abgestimmt sind. Attribute die in Attr definiert sind, werden automatisch in AttrMask hinzugefügt. Geben Sie Attribute an, die nicht in AttrMask eingesetzt werden sollen. Das Pseudo-Attribut RTF_FIND_NO_ALIAS hat keine Auswirkung auf AttrMask.
pResult	INT	Zeiger auf einen int Wert, der mit dem Operationsergebnis gefüllt wird. Bei Erfolg wird dieser Wert größer als oder gleich 0 sein. Wenn der Vorgang fehlschlagen ist, wird ein negativer Fehler-Code zurückgegeben.
fileInfo	_DDE_INFO	Zeiger auf eine _DDE_INFO Struktur. Wenn die Funktion erfolgreich ist und pResult auch Erfolg anzeigt, wird diese Struktur mit der Verzeichniseintragung der gefundenen Datei gefüllt. Dieser Parameter kann NULL werden.
filename	CHAR	Zeiger auf einen String Puffer, der den Dateinamen ohne Pfad empfängt, wenn eine Datei gefunden wird.
maxLength	UINT	Größe des von FileName angezeigten Puffers in Byte. Dateinamen mit einer Länge größer als maxLength -1 werden nicht gefunden.

Rückgabeparameter	Typ	Beschreibung
		<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>



Es ist wichtig `LslFindFirstH()`/`LslFindNextH()` Anrufe mit `LslFindCloseH()` zu schließen, wenn Informationen nicht mehr benötigt werden. `LslFindFirstH()` schließt einen vorhergehenden `LslFindFirstH()` Aufruf. Das Betriebssystem unterstützt nur ein offenes `LslFindFirstH()`.

6.1.11.18 `LslFindFirst`, `LslFindNext`, `LslFindClose`

Entsprechen den Funktionen `LslFindFirstH()`, `LslFindNextH()` und `LslFindCloseH()` mit dem Parameter `ocbNum = 0`.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFindFirst(
const char          *namePattern,
    uint8_t          attr,
    uint8_t          attrMask,
    int32_t          *pResult,
    _DDE_INFO        *fileInfo,
    char             *fileName,
    uint32_t          maxLength
);

extern "C" LSL_BOOL LASAL32_EXPORTS LslFindNext(
    int32_t          *pResult,
    _DDE_INFO        *fileInfo,
    char             *fileName,
    uint32_t          maxLength
);

extern "C" LSL_BOOL LASAL32_EXPORTS LslFindClose(
    int32_t          *pResult
);
```

6.1.11.19 `LslFindNextH`

Die Funktion `LslFindNextH()` findet andere Dateien mit den gleichen Suchkriterien wie ein vorhergehender Aufruf zu [LslFindFirst\(\)](#).

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFindNextH(
    int32_t          ocbNum,
    int              *pResult,
    _DDE_INFO        *fileInfo,
```

```

char      *fileName,
UINT      maxLength
);

```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
pResult	int	Zeiger auf einen INT Wert, der mit dem Operationsergebnis gefüllt wird.	
		<div style="display: flex; justify-content: space-around; align-items: center;"> ≥0 Erfolg </div> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 5px;"> <0 Negativer Fehler-Code </div>	
fileInfo	_DDE_INFO	Zeiger auf eine _DDE_INFO Struktur. Wenn die Funktion erfolgreich ist und pResult auch Erfolg anzeigt, wird diese Struktur mit der Verzeichniseintragung der gefundenen Datei gefüllt. Dieser Parameter kann NULL werden.	
Filename	char	Zeiger auf einen String Puffer, der den Dateinamen ohne Pfad empfängt, wenn eine Datei gefunden wird.	
maxLength	UINT	Größe des vom FileName angezeigten Puffers in Bytes. Dateinamen mit einer Länge größer als maxLength -1 werden nicht gefunden.	
Rückgabeparameter	Typ	Beschreibung	
		<div style="display: flex; justify-content: space-around; align-items: center;"> TRUE Funktion erfolgreich </div> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 5px;"> FALSE Fehler </div>	
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	

Es ist wichtig die geöffneten Funktionen [LslFindFirstH\(\)](#)/[LslFindNextH\(\)](#) zu schließen, wenn keine Informationen mehr benötigt werden.



6.1.11.20 LslFileTransfer

Entspricht der Funktion [LslFileTransferH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransfer(
    uint32_t      direction,
    const char    *fileSrc,
    const char    *fileDest,
    uint32_t      flags,

```

```
CB_FUNCTYPE2  *callback,
void          *pUser
);
}
```

6.1.11.21 LslFileTransferFlush

Entspricht LslFileTransferFlushH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferFlush(void);
```

6.1.11.22 LslFileTransferFlushH

Löst einen Cache-Flush des Dateisystems auf der SPS aus. Verwenden Sie diese Funktion nach Dateidownloads mit der Funktion LslFileTransferH() wenn die Option LSL_FILETRANSFER_DONT_USE_AUTO_FLUSH gesetzt ist.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferFlushH(
    int32_t  ocbNum
);
```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
Rückgabeparameter	Typ	Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
Mit LslGetError kann der letzte Fehlercode ermittelt werden. Dieser Code entspricht entweder einem Lasal32- oder einem systemspezifischen Fehlercode (Windows: GetLastError oder Linux: errno).			

6.1.11.23 LslFileTransferH

Die Funktion LslFileTransferH() überträgt Dateien zwischen dem PC und der Steuerung. Im Gegensatz zur Funktion [FileSaveH\(\)](#) weist die Funktion LslFileTransferH() ein fehlertoleranteres Verhalten bei qualitativ minderwertigen Verbindungen auf. Für neue Projekte sollte nur noch die Funktion [LslFileTransfer\(\)](#) verwendet werden. Die Funktion meldet einen SPS-Fehler-Code im Fall eines Remote-Errors.

```
typedef uint32_t __cdecl CB_FUNCTYPE2(
    void          *pData,
    uint32_t      val,
    uint32_t      state
);
```

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferH(
    int32_t          ocbNum,
    uint32_t         direction,
    const char      *fileSrc,
    const char      *fileDest,
    uint32_t         flags,
    CB_FUNCTYPE2    *callback,
    void            *pUser
);

```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
direction	uint32_t	Gibt die Richtung an, in der die Datei übertragen werden soll (von der Steuerung oder zu der Steuerung).	
		Attribute	Bedeutung
		LSL_FILETRANSFER_DIRECTION_TO PLC	Die Datei wird vom PC auf die Steuerung übertragen.
		LSL_FILETRANSFER_DIRECTION_FROMPLC	Die Datei wird von der Steuerung auf den PC übertragen.
fileSrc	const char	Zeiger auf den Namen der Quelldatei	
fileDest	const char	Zeiger auf den Namen der Zieldatei	
flags	uint32_t	Attribute	Bedeutung
		LSL_OPENFLAGS_RDONLY	Öffnet die Datei nur zum Lesen
		LSL_OPENFLAGS_WRONLY	Öffnet die Datei nur zum Schreiben
		LSL_OPENFLAGS_CREATE	Öffnet die Datei lesend und schreibend. Die Datei wird angelegt falls sie noch nicht existiert.
		LSL_FILETRANSFER_DONT_USE_AUTO_FLUSH	Option zum schnellen Herunterladen vieler Dateien oder Verzeichnisse mit LslFileTransferH() vom PC zur SPS, indem automatische Dateisystem-Cache-Flushes verhindert werden. Nachdem alle Dateien heruntergeladen sind, wird ein Aufruf von LslFileTransferFlushH() empfohlen.

		<p>Wenn das Betriebssystem diese Option nicht unterstützt, schlägt LslFileTransferH() NICHT fehl, sondern führt automatische Cache-Flushes durch.</p>
callback	CB_FUNCTYPE2*	<p>Zeiger auf eine CB_FUNCTYPE2 Callback-Funktion. Bei der Übertragung der Datei ist es möglich, dass die Daten in Blöcke unterteilt werden. Die Callback-Funktion wird vor der Übertragung der Datei einmal mit der Dateilänge in Byte und einmal mit der Anzahl der verbleibenden Blöcke als Parameter aufgerufen. Die Callback-Funktion wird während der Übertragung der Datei mit der Anzahl der Byte, die schon kopiert wurden als Parameter aufgerufen.</p> <p>Die Callback-Funktion kann verwendet werden, um z.B. einen Fortschrittsbalken zu aktualisieren. Wenn der Parameter callback NULL ist, wird keine Callback-Funktion aufgerufen.</p>
pUser	void	Anwender-Zeiger, welcher an die Callback-Funktion übergeben wird (falls verwendet)
Rückgabeparameter	Typ	Beschreibung
		<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>

Anforderungen

Lasal32.dll ab 01.01.109

Beispiel

```
uint32_t __cdecl NewCallbackString(void *pData, uint32_t val, uint32_t state)
{
    const char *pText = (const char *) pData;
    if (state == CBSTATE_INIT)
        printf("%s Init %u\n", pText, val);
    else if (state == CBSTATE_STATE_BEGIN)
        printf("%s Begin: %u\n", pText, val);
    else if (state == CBSTATE_STATE)
        printf("%s Do: %u\n", pText, val);
    else if (state == CBSTATE_FINISHED)
        printf("\n%s Finished %u\n", pText, val);
    else if (state == CBSTATE_ERROR)
        printf("\n%s Error:0x%08X, %u\n", pText, val, val);
```

```

else if (state == CBSTATE_CANCELED)
    printf("\n%s canceled\n", pText);
return CBRETURN_CONTINUE;
}

if (::Online(strConnection, 0, 0, 0, 0))
{
    printf("Testing FileTransfer to PLC\n");
    if (LslFileTransfer(LSL_FILETRANSFER_DIRECTION_TOPLC, "C:\\test.txt",
        "C:\\test.txt", LSL_OPENFLAGS_CREATE | LSL_OPENFLAGS_WRONLY,
        NewCallbackString, (void*)"FileTransfer"))
    {
        printf("FileTransfer ToPLC OK\n");

        printf("Testing FileTransfer from PLC\n");
        if (LslFileTransfer(LSL_FILETRANSFER_DIRECTION_FROMPLC, "C:\\test.txt",
            "C:\\test.001", LSL_OPENFLAGS_CREATE | LSL_OPENFLAGS_WRONLY,
            NewCallbackString, (void*)"FileTransfer"))
        {
            printf("FileTransfer FromPLC OK\n");
        }
        else
            printf("FileTransfer FromPLC Error:0x%08X, %u\n", GetLastError(),
                GetLastError());
    }
    else
        printf("FileTransfer ToPLC Error:0x%08X, %u\n", GetLastError(), GetLastError());
}
else
    printf("Error Online:0x%08X \n", GetLastError());
::Offline();

```

6.1.11.24 LslFileTransferFromPlcBuf

Entspricht der Funktion [LslFileTransferFromPlcBufH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferFromPlcBuf(
    const char      *fileSrc,
    uint8_t         *destBuf,
    uint32_t        size,
    uint32_t        *getsize,
    CB_FUNCTYPE2    *callback,
    void           *pUser
);

```

6.1.11.25 LslFileTransferFromPlcBufH

Die Funktion `LslFileTransferFromPlcBufH()` kann verwendet werden, um den Inhalt einer Datei auf der Steuerung direkt in einen Puffer zu kopieren.

```
extern "C" LSL_BOOL WNAPI LslFileTransferFromPlcBuf(
    int32_t          ocbNum,
    Const char       *fileSrc,
    Uint8_t          *destBuf,
    Uint32_t          size,
    Uint32_t          *getsize,
    CB_FUNCTYPE2     *callback,
    Void             *pUser
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
FileSrc	CONST CHAR	Zeiger auf den Dateinamen auf der Steuerung
DestBuf	uint8_t	Zeiger auf Zielpuffer für eingehende Daten
Size	uint32_t	Größe des Zielpuffers
Getsize	uint32_t*	Optionaler Zeiger auf eine Variable, in der die Gesamtzahl der übertragenen Byte gespeichert wird
callback	CB_FUNCTYPE2*	<p>Zeiger auf eine CB_FUNCTYPE2 Callback-Funktion. Bei der Übertragung der Datei ist es möglich, dass die Daten in Blöcke unterteilt werden. Die Callback-Funktion wird vor der Übertragung der Datei einmal mit der Dateilänge in Byte und einmal mit der Anzahl der verbleibenden Blöcke als Parameter aufgerufen. Die Callback-Funktion wird während der Übertragung der Datei mit der Anzahl der Byte, die schon kopiert wurden als Parameter aufgerufen.</p> <p>Die Callback-Funktion kann verwendet werden, um z.B. einen Fortschrittsbalken zu aktualisieren. Wenn der Parameter <code>callback</code> <code>NULL</code> ist, wird keine Callback-Funktion aufgerufen.</p>
pUser	void*	Ein optionaler Anwender-Zeiger, welcher an die Callback-Funktion übergeben wird
		Rückgabeparameter
		<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>

Anforderungen

Lasal32.dll ab 01.01.109

Beispiel

```
uint32_t __cdecl NewCallbackString(void *pData, uint32_t val, uint32_t state)
{
    const char *pText = (const char *) pData;
    if (state == CBSTATE_INIT)
        printf("%s Init %u\n", pText, val);
    else if (state == CBSTATE_STATE_BEGIN)
        printf("%s Begin: %u\n", pText, val);
    else if (state == CBSTATE_STATE)
        printf("%s Do: %u\n", pText, val);
    else if (state == CBSTATE_FINISHED)
        printf("\n%s Finished %u\n", pText, val);
    else if (state == CBSTATE_ERROR)
        printf("\n%s Error:0x%08X, %u\n", pText, val, val);
    else if (state == CBSTATE_CANCELLED)
        printf("\n%s canceled\n", pText);
    return CBRRETURN_CONTINUE;
}

if (::Online("10.10.170.190", 0, 0, 0, 0))
{
    uint32_t nReadSize;
    LSL_FILE_INFO fileInfo;

    if (FileInfo((char *)&fileInfo, "C:\\\\test.txt", sizeof(fileInfo)))
    {
        void* pBuff = malloc(fileInfo.lFileSize);

        printf("Testing File from PLC\\n");
        if (LslFileTransferFromPlcBuf("C:\\\\test.txt", (uint8_t*)pBuff,
            fileInfo.lFileSize, &nReadSize, NewCallbackString, (void*)"FileTransfer"))
        {
            printf("LslFileTransferFromPLC (%u Byte, BufSize:%u)\\n", nReadSize,
                fileInfo.lFileSize);

            printf("Testing File to PLC\\n");

            if (LslFileTransferToPlcBuf(pBuff, fileInfo.lFileSize, "C:\\\\test.001",
                LSL_OPENFLAGS_CREATE | LSL_OPENFLAGS_WRONLY, NewCallbackString,
                (void*)"FileTransfer"))
            {
                printf("LslFileTransferToPLC OK\\n");
            }
            else
                printf("LslFileTransferToPLC Error:0x%08X, %u\\n", GetLastError(),
                    GetLastError());
        }
    }
}
```

```

    }
    else
    {
        printf("LslFileTransferFromPLC Error:0x%08X, %u\n", GetLastError(),
               GetLastError());
    }
    free(pBuff);
}
else
    printf("FileInfo failed 0x%08X\n", GetLastError());
}
else
    printf("Error Online:0x%08X \n", GetLastError());
::Offline();

```

6.1.11.26 LslFileTransferMakeDir

Entspricht LslFileTransferMakeDirH() mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferMakeDir(
    const char*  dirName,
    uint32_t      flags
);

```

6.1.11.27 LslFileTransferMakeDirH

Diese Funktion erstellt ein Verzeichnis mit dem angegebenen Pfad. Der Befehl schlägt fehl, wenn er vom Betriebssystem nicht unterstützt wird.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferMakeDirH(
    int32_t      ocbNum,
    const char*  dirName,
    uint32_t      flags
);

```

Übergabeparameter		Typ	Beschreibung
ocbNum		int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
dirName		const char*	Zeiger auf einen 0-terminierten String, der den vollständigen Pfad angibt, z.B.: C:\temp\test1\result
flags		uint32_t	Bit 0 (recursive): Verzeichnisse rekursiv erstellen (1) oder nicht (0). Bit 1 (dontSync): Alle nicht leeren Puffer im Kernel werden auf das Gerät geschrieben (0) oder nicht (1)
Rückgabeparameter		Typ	Beschreibung
			TRUE Funktion erfolgreich

		FALSE Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.		

Beispiel

```
//recursiv, don't sync. Alle drei Verzeichnisse werden nacheinander erstellt.
if (LslFileTransferMakeDir("C:\\\\testdir\\\\subdir1\\\\subsubdir1", 3) == FALSE)
    printf("LslFileTransferMakeDir() failed(0x%08X)\\n", GetLastError());
```

6.1.11.28 LslFileTransferToPlcBuf

Entspricht der Funktion [LslFileTransferToPlcBufH\(\)](#) mit ocbNum = 0.

```
LSL_BOOL LASAL32_EXPORTS LslFileTransferToPlcBuf(
    const void    *srcData,
    uint32_t      size,
    const char   *fileDest,
    uint32_t      flags,
    CB_FUNCTYPE2  *callback,
    void          *pUser
);
```

6.1.11.29 LslFileTransferToPlcBufH

Die Funktion [LslFileTransferToPlcBufH\(\)](#) kann verwendet werden, um Daten auf die Steuerung zu übertragen. Diese Daten werden direkt in die angegebene Datei geschrieben und es muss keine lokale Datei erstellt werden.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferToPlcBufH(
    int32_t      ocbNum,
    const void   *srcData,
    uint32_t      size,
    const char   *fileDest,
    uint32_t      flags,
    CB_FUNCTYPE2  *callback,
    void          *pUser
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
SrcData	CONST VOID	Zeiger auf die zu übertragenen Daten.
Size	uint32_t	Datenlänge
FileDest	CONST CHAR	Zeiger auf den Dateinamen auf der Steuerung

Flags	uint32_t	Datei-Flags				
Callback	CB_FUNCTYPE2*	<p>Zeiger auf eine CB_FUNCTYPE2 Callback-Funktion. Bei der Übertragung der Datei ist es möglich, dass die Daten in Blöcke unterteilt werden. Die Callback-Funktion wird vor der Übertragung der Datei einmal mit der Dateilänge in Byte und einmal mit der Anzahl der verbleibenden Blöcke als Parameter aufgerufen. Die Callback-Funktion wird während der Übertragung der Datei mit der Anzahl der Byte, die schon kopiert wurden als Parameter aufgerufen.</p> <p>Die Callback-Funktion kann verwendet werden, um z.B. einen Fortschrittsbalken zu aktualisieren. Wenn der Parameter callback NULL ist, wird keine Callback-Funktion aufgerufen.</p>				
PUser	VOID	Optionaler Anwender-Zeiger welcher in die Callback-Funktion hineingereicht wird				
Rückgabeparameter	Typ	Beschreibung				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50px; text-align: center;">TRUE</td><td>Funktion erfolgreich</td></tr> <tr> <td style="text-align: center;">FALSE</td><td>Fehler</td></tr> </table> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>	TRUE	Funktion erfolgreich	FALSE	Fehler
TRUE	Funktion erfolgreich					
FALSE	Fehler					

Anforderungen

Lasal32.dll ab 01.01.109

Beispiel

Siehe [LslFileTransferFromPlcBufH\(\)](#).

6.1.11.30 LslFormatDrive

Entspricht der Funktion [LslFormatDriveH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFormatDrive(
    uint8_t          ucDriveLetter,
    uint32_t         ulOption,
    int32_t          *pResult,
    CB_FORMAT_FUNCTYPE  FormatInfoCallback
);
```

6.1.11.31 LslFormatDriveH

Die Funktion LslFormatDriveH() formatiert ein angegebenes Laufwerk.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFormatDriveH(
    int32_t          ocbNum,
    unsigned char    ucDriveLetter,
    unsigned long    ulOption,
    int              *pResult,
    CB_FORMAT_FUNCTYPE FormatInfoCallback
);
```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
ucDriveLetter	UNSIGNED CHAR	Laufwerkbuchstabe der zu formatierenden Festplatte.				
ulOption	UNSIGNED LONG	Reserviert				
pResult	INT	<p>Zeiger auf einen INT Wert, der mit dem Operationsergebnis gefüllt wird</p> <table border="1" style="margin-left: 20px;"> <tr> <td>≥0</td> <td>Erfolg</td> </tr> <tr> <td><0</td> <td>Negativer Fehler-Code</td> </tr> </table>	≥0	Erfolg	<0	Negativer Fehler-Code
≥0	Erfolg					
<0	Negativer Fehler-Code					
FormatInfoCallback	CB_FORMAT_FUNC_TYPE	Zeiger auf eine CB_FORMAT_FUNCTYPE Callback Funktion, die Informationen über den Formatierungsfortschritt zeigt. Dieser Parameter kann NULL sein, wenn keine Informationen erforderlich sind.				
Übergabeparameter	Typ	Beschreibung				
		<table border="1" style="margin-left: 20px;"> <tr> <td>TRUE</td> <td>Funktion erfolgreich</td> </tr> <tr> <td>FALSE</td> <td>Fehler</td> </tr> </table> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>	TRUE	Funktion erfolgreich	FALSE	Fehler
TRUE	Funktion erfolgreich					
FALSE	Fehler					

Die benötigte Ausführungszeit hängt von dem vorhandenen Speicherplatz des Mediums ab. Für sehr großen Speichermedien kann die Funktion bis mehrere Minuten dauern.



6.1.11.32 LslGetDiskSpace

Entspricht der Funktion [LslGetDiskSpaceH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDiskSpace(
    const char    *ccDrive,
    uint32_t      *pBytesPerSector,
    uint32_t      *pSectorsPerCluster,
    uint32_t      *pTotalClusters,
    uint32_t      *pFreeClusters,
    uint32_t      ulOption,
    int32_t       *pResult
);
```

6.1.11.33 LslGetDiskSpaceH

Funktion, um Informationen von einem bestimmten Laufwerk abzufragen, die zur Berechnung des benötigten und des freien Festplattenspeicherplatzes benötigt werden.

```
LSL_BOOL LASAL32_EXPORTS LslGetDiskSpaceH(
    int32_t      ocbNum,
    const char   *ccDrive,
    uint32_t      *pBytesPerSector,
    uint32_t      *pSectorsPerCluster,
    uint32_t      *pTotalClusters,
    uint32_t      *pFreeClusters,
    uint32_t      ulOption,
    int32_t       *pResult
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
ccDrive	CONST CHAR	Zeiger auf den Laufwerknamen
pBytesPerSector	uint32_t	Zeiger auf eine Variable, die die Anzahl der Bytes pro Sektor enthält
PSectorsPerCluster	uint32_t	Zeiger auf eine Variable, die die Anzahl der Sektoren pro Cluster enthält
pTotalClusters	uint32_t	Zeiger auf eine Variable für die Anzahl der Cluster
pFreeCluster	uint32_t	Zeiger auf eine Variable, die die Anzahl der freien Cluster enthält
ulOption	uint32_t	Reserviert
pResult	int32_t	Zeiger auf einen int Wert, der mit dem Operationsergebnis gefüllt wird
		≥0 Erfolg

		<0 Negativer Fehler-Code
Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich
		FALSE Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.		

Anforderungen

Lasal32.dll	ab 01.01.019
LasalOS	ab 01.01.079

Beispiel

```

BOOL rc;
int iRC;

unsigned long BpS;
unsigned long SpC;
unsigned long TC;
unsigned long FC;

rc = LslGetDiskSpace("C:\\\\", &BpS, &SpC, &TC, &FC, 0, &iRC);
if (rc == false)
    ; // command not successful
else
{
    ; // command successful

    if (iRC < 0)
        ; // operation not successful

    else
    {
        printf("Diskspace: %d\n", Bps * SpC * TC);
        printf("Freespace: %d\n", Bps * SpC * FC);
    }
}

```

6.1.11.34 LslGetDriveListShort

Entspricht der Funktion [LslGetDriveListShortH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDriveListShort(
    uint32_t *pDriveList,
    uint32_t ulSize,
    uint32_t ulOption,
    int32_t *pResult
);
```

6.1.11.35 LslGetDriveListShortH

Funktion, um Informationen über die verfügbaren Laufwerke sowie dessen Eigenschaften zu erhalten.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDriveListShortH(
    int32_t ocbNum,
    uint32_t *pDriveList,
    uint32_t ulSize,
    uint32_t ulOption,
    int32_t *pResult
);
```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
pDriveList	uint32_t	Zeiger auf ein unsigned long Array für Informationen über die Laufwerke				
ulSize	uint32_t	Größe des Arrays für die Informationen über die Laufwerke				
ulOption	uint32_t	Option, welche Informationen ausgelesen werden sollen				
pResult	int32_t	Zeiger auf einen int Wert, der mit dem Operationsergebnis gefüllt wird <table border="1" data-bbox="459 1015 1008 1110"> <tr> <td>≥0</td> <td>Erfolg</td> </tr> <tr> <td><0</td> <td>Negativer Fehler-Code</td> </tr> </table>	≥0	Erfolg	<0	Negativer Fehler-Code
≥0	Erfolg					
<0	Negativer Fehler-Code					
Rückgabeparameter	Typ	Beschreibung				
		<table border="1" data-bbox="459 1158 1008 1253"> <tr> <td>TRUE</td> <td>Funktion erfolgreich</td> </tr> <tr> <td>FALSE</td> <td>Fehler</td> </tr> </table> Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	TRUE	Funktion erfolgreich	FALSE	Fehler
TRUE	Funktion erfolgreich					
FALSE	Fehler					

Auf einem typischen System sind 26 Laufwerke (A-Z) verfügbar. Das Array sollte dafür über mindestens 26 Elemente verfügen, um alle möglichen Informationen zu erhalten.

Mögliche Werte für die Parameter ulOption, die "OR kombiniert" werden können, sind:

```
#define DRIVELIST_OPT_MOUNTED      0x01 Nur eingebaute Geräte.
#define DRIVELIST_OPT_MOUNTSTATE   0x02 Der Status eines eingebauten Gerätes.
#define DRIVELIST_OPT_MOUNTED      DRIVELIST_OPT_MOUNTED muss gesetzt werden.
#define DRIVELIST_OPT_TYPE         0x04 Gerätetyp
#define DRIVELIST_OPT_DRIVER       0x08 Der angegebene Gerätetreiber.
```

Die Information eines jeden Laufwerks ist der Index des Arrays. Index 0 wird als Laufwerk 'A' angegeben, Index -1 wird Laufwerk 'B', usw.

Jeder Index enthält einen 32-Bit Wert, der Informationen über das Laufwerk, abhängig von dem Parameter ulOption, enthält.

Information Flags

```
// General
#define DRIVELIST_FLOPPY           0x00000001
#define DRIVELIST_FDISK            0x00000002
#define DRIVELIST_MOUNTED          0x00000004
// Mountstate
#define DRIVELIST_MS_INITIALIZED   0x00000100
#define DRIVELIST_MS_MOUNTED      0x00000200
#define DRIVELIST_MS_ACCESSIBLE   0x00000400
#define DRIVELIST_MS_HASFILESYSTEM 0x00000800
// Driver
#define DRIVELIST_DRV_FLOPPY       0x00010000
#define DRIVELIST_DRV_IDE          0x00020000
#define DRIVELIST_DRV_USB          0x00030000
#define DRIVELIST_DRV_CME221       0x00040000
#define DRIVELIST_DRV_SMARTMEDIA   0x00050000
#define DRIVELIST_DRV_UDRAM        0x00060000
// Masks
#define DRIVELIST_GI_MASK          0x00000007
#define DRIVELIST_MS_MASK          0x0000F000
#define DRIVELIST_DRIVER_MASK      0x00070000
```

Wenn ein Index mit 0 gefüllt ist, ist kein Laufwerk verfügbar oder es gibt keine Informationen für die angegebene Option, die an die Funktion innerhalb der ulOption Parameter übertragen wurde.

Anforderungen

Lasal32.dll	ab 01.01.019
LasalOS	ab 01.01.079

Beispiel

```
int iRC, i;
BOOL rc;
unsigned long DriveList[26];
unsigned long ulOption = DRIVELIST_OPT_MOUNTED |
                        DRIVELIST_OPT_MOUNTSTATE |
                        DRIVELIST_OPT_TYPE |
                        DRIVELIST_OPT_DRIVER;

memset(DriveList, 0, sizeof(DriveList));

rc = LslGetDriveListShortH(0,DriveList, sizeof(DriveList), ulOption, &iRC);

ShowDriveListOnScreen(DriveList, ulOption);

void ShowDriveListOnScreen(unsigned long * pDriveList,
                           unsigned long ulOption)

{
    int i;

    printf("Drive MountState      Type      Driver\n");
    printf("-----\n");

    for (i = 0; i < 26; i++)
    {
        if (pDriveList[i] == 0)
        {
            printf("-----\n");
            continue;
        }

        if (ulOption & DRIVELIST_OPT_MOUNTED)
        {
            if (pDriveList[i] & DRIVELIST_MOUNTED)
            {
                printf(" %c:  ", 'A' + i);

                if (ulOption & DRIVELIST_OPT_MOUNTSTATE)
                {
                    if (pDriveList[i] & DRIVELIST_MS_INITIALIZED)
                        printf("Initialized      ");
                    else if (pDriveList[i] & DRIVELIST_MS_MOUNTED)
                        printf("Mounted      ");
                    else if (pDriveList[i] & DRIVELIST_MS_ACCESSIBLE)
                        printf("Accessible      ");
                    else if (pDriveList[i] & DRIVELIST_MS_HASFILESYSTEM)
                        printf("HasFileSystem      ");
                }
            }
            else
                printf("-----  ");
        }
    }
}
```

```
        }
        else
            printf("      Not mounted    ");
    }
else
    printf("----- ----- ");

if (ulOption & DRIVELIST_OPT_TYPE)
{
    if (pDriveList[i] & DRIVELIST_FLOPPY)
        printf("Floppy ");
    else if (pDriveList[i] & DRIVELIST_FDISK)
        printf("FDISK ");
}
else
    printf("----- ");

if (ulOption & DRIVELIST_OPT_DRIVER)
{
    if ((pDriveList[i] & DRIVELIST_DRIVER_MASK) == DRIVELIST_DRV_FLOPPY)
        printf("Floppy");
    else if ((pDriveList[i] & DRIVELIST_DRIVER_MASK) == DRIVELIST_DRV_IDE)
        printf("IDE");
    else if ((pDriveList[i] & DRIVELIST_DRIVER_MASK) == DRIVELIST_DRV_USB)
        printf("USB");
    else if ((pDriveList[i] & DRIVELIST_DRIVER_MASK) ==
              DRIVELIST_DRV_CME221)
        printf("CME221");
    else if ((pDriveList[i] & DRIVELIST_DRIVER_MASK) ==
              DRIVELIST_DRV_SMARTMEDIA)
        printf("SmartMedia");
    else if ((pDriveList[i] & DRIVELIST_DRIVER_MASK) ==
              DRIVELIST_DRV_UDRAM)
        printf("UDRAM");
    else
        printf("Unknown");
}
else
    printf("----- ");

    printf("\n");
}
}
```

6.1.11.36 LslRenameFileDir

Entspricht der Funktion [LslRenameFileDirH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRenameFileDir(
    const char  *ccFileDialogName,
    const char  *ccNewName,
    uint32_t     ulOption,
```

```
    int32_t      *pResult
);
}
```

6.1.11.37 LslRenameFileDirH

Die Funktion LslRenameFileDirH() wird verwendet, um eine Datei oder ein Verzeichnis umzubenennen.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRenameFileDirH(
    int32_t      ocbNum,
    const char   *ccFileDirectoryName,
    const char   *ccNewName,
    uint32_t      ulOption,
    int32_t      *pResult
);
```

Übergabeparameter	Typ	Beschreibung				
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())				
ccFileDirectoryName	CONST CHAR	Zeiger auf einen umzubenennenden Datei- oder Verzeichnisnamen				
ccNewName	CONST CHAR	Zeiger auf den neuen Namen				
ulOption	uint32_t	Reserviert				
pResult	int32_t	<p>Zeiger auf einen int Wert, der mit dem Operationsergebnis gefüllt wird</p> <table border="1" style="margin-left: 20px;"> <tr> <td style="background-color: #e0f2f1;">≥0</td> <td>Erfolg</td> </tr> <tr> <td style="background-color: #e0f2f1;"><0</td> <td>Negativer Fehler-Code</td> </tr> </table>	≥0	Erfolg	<0	Negativer Fehler-Code
≥0	Erfolg					
<0	Negativer Fehler-Code					
Rückgabeparameter	Typ	Beschreibung				
		<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>				

Anforderungen

Lasal32.dll	ab 01.01.019
LasalOS	ab 01.01.079

Beispiel

```
BOOL rc;
int iRC;

rc = LslRenameFileDirH(0, "C:\autoexec.lsl", "C:\autoexec.bak", 0, &iRC);
if (rc == false)
; // command not successful
else
; // command successful, iRC for information of the operation
```

6.1.11.38 Parameter einer CB_FUNCTYPE Callback-Funktion

Übergabeparameter	Typ	Beschreibung	
Val		Callback-Funktion wird vor der Übertragung eines Blocks mit der Anzahl der verbleibenden Blöcke als Parameter aufgerufen	
Rückgabeparameter		CBRETURN_CONTINUOUS (0)	Transfer fortsetzen
		CBRETURN_ABORT (1)	Transfer abbrechen

6.1.11.39 Parameter einer CB_FUNCTYPE2 Callback-Funktion

Übergabeparameter	Typ	Beschreibung	
PData	void	Zeiger pUser, der an die Dateiübertragungsfunktion (LslFileTransfer(), LslFileTransferFrom PlcBuf(), LslFileTransferToPlcBuf(), LslDownloadOSH()) übergeben wurde	
Val	uint32_t	Abhängig vom Parameter State werden hier diverse Werte übergeben. Siehe auch Parameter State.	
State	uint32_t	Gibt den aktuellen Status der Dateiübertragung an. In der folgenden Tabelle sind alle möglichen Zustände aufgeführt.	
		CBSTATE_INIT	Der Datei-Transfer startet und in Val steht die Anzahl der Byte, welche insgesamt übertragen werden (Dateilänge).
		CBSTATE_STATE_BEGIN	Der Datei-Transfer startet und in Val steht die Anzahl der Datenblöcke, welche insgesamt übertragen werden.

		<p>CBSTATE_STATE Der Datei-Transfer läuft. In Val steht die Anzahl der Byte, die schon kopiert wurden.</p> <p>CBSTATE_FINISH Der Datei-Transfer ist abgeschlossen. In Val steht die Anzahl der Byte, die insgesamt übertragen wurden.</p> <p>CBSTATE_ERROR Es ist ein Fehler aufgetreten. Der Parameter Val enthält eine Fehlernummer.</p> <p>CBSTATE_CANCEL Die Callback-Funktion hat den Datei-Transfer abgebrochen.</p>
Rückgabeparameter	Typ	Beschreibung
	uint32_t	<p>Die Callback-Funktion kann folgende Werte zurückgeben:</p> <p>CBRETURN_CONTI Datei-Transfer fortsetzen NUE</p> <p>CBRETURN_ABOR Datei-Transfer abbrechen T</p>

6.1.12 Verwalten der RefreshListe

Eine Refresh-Liste wird zum optimierten Datenaustausch zwischen einer SPS und einer externen Station (z.B. PC + Windows) verwendet. Bei einer Refresh-Liste können LASAL-Server und globale Variablen angemeldet werden. Um Objekt-Member-Variablen anzumelden, muss die Applikations-SW selbst die Adresse beschaffen. Siehe unten und Kapitel 8, Datenaustausch mit Objekt-Member-Variablen. Anschließend wird jede Änderung der Daten (Werte) eines angemeldeten Elementes dem Aufrufer (Callback) mitgeteilt. Somit ist die Belastung der Kommunikation auf ein Minimum reduziert und der Anwender kann gezielt auf eine Datenänderung reagieren. Es ist also kein ständiges Abfragen auf Datenänderungen mehr notwendig.

Anschließend werden alle Funktionen zum Erstellen, Verwalten und Löschen einer Refresh-Liste erklärt und mit kleinen Beispielen veranschaulicht.

Vom System (SPS) aus stehen 2 Refresh-Listen zur Verfügung. Diese beiden Listen werden in der SPS vollständig getrennt abgearbeitet und stehen dem Anwender zur freien Verfügung. Zur besseren Unterscheidung sind folgende Konstanten definiert.

```
#define LSL_RL_FLAG_DYNAMIC_LIST      0x00000000
#define LSL_RL_FLAG_PERMANENT_LIST     0x00000001
```

Zum Beispiel kann die LSL_RL_FLAG_DYNAMIC_LIST für temporäre, nicht ständig zu überwachende Werte verwendet werden. In der LSL_RL_FLAG_PERMANENT_LIST hingegen befinden sich nur Einträge deren Werte ständig benötigt werden.

Definition der notwendigen Konstanten und Struktur zur Datenbereitstellung

```
#define LSL_RL_Type_SERVER           0x00000000
#define LSL_RL_Type_STRINGSERVER     0x00000001
#define LSL_RL_Type_CLIENT           0x00000002
#define LSL_RL_Type_GLOBAL            0x00000003

#define LSL_RL_Type_MASK             0x0000000F

#pragma pack(push, 1)
struct SRLVarInfo
{
    DWORD dwAddr;    // Adresse der Variable oder Server
    DWORD dwType;    // Eintragungstype: LSL_RL_TYPE...
    DWORD dwSize;
};

#pragma pack(pop)
```

Refresh-Liste erzeugen

1. Zuerst muss die Refresh-Liste mit LslRefreshListCreateExt() erzeugt werden. Dabei wird auch die Verbindung zur Refresh-Liste hergestellt.
2. Um numerische Server (Integer und REAL), String-Server und globale Variablen (Integer, REAL, LREAL, ARRAY und STRUCT) bei der Refresh-Liste anzumelden, werden für jede Variable die Funktionen LslRefreshListGetVarInfo() und LslRefreshListAdd() aufgerufen. Hinweis: vor dem Aufruf von LslRefreshListAdd() muss in SRLVarInfo.VarInfo.dwSize die Größe der Variablen in Byte eingetragen werden. Bei großen Arrays und Strukturen muss der Parameter dwRefreshTime auf einen entsprechend großen Wert gesetzt werden. Achtung: bei Strukturen muss das Alignment am PC und auf der SPS gleich sein.
3. Um Objekt-Member-Variablen (oder mit malloc() allokierte Speicherbereiche) bei der Refresh-Liste anzumelden, muss zuerst die Adresse eines Adressen-Servers mit LslRefreshListGetVarInfo(iRLB, "Objektname.AdressenServername", &VarInfo) gelesen werden. Anschließend kann mit LslRefreshListGetData(iRLB, &VarInfo, (uint8_t*)&VarInfo.dwAddr, sizeof(plcptr_t)) die Adresse der Objekt-Member-Variablen bzw. des Speicherbereichs vom Adressen-Server gelesen werden. Hinweis: vor dem Aufruf von LslRefreshListAdd() müssen in SRLVarInfo.VarInfo folgende Elemente versorgt werden: dwAddr=ermittelte Adresse, dwSize=Größe der Variable in Byte, dwType= LSL_RL_TYPE_GLOBAL. Bei großen Arrays und Strukturen muss der

Parameter dwRefreshTime auf einen entsprechend großen Wert gesetzt werden. Achtung: bei Strukturen muss das Alignment am PC und auf der SPS gleich sein.

4. Wenn die Refresh-Liste fertig ist, kann die Überwachung der darin enthaltenen Variablen mit der Funktion LslRefreshListStart() gestartet werden. Ab jetzt wird die in Punkt 1 angegebene Callback-Funktion aufgerufen, wenn sich der Wert einer Variablen in der Refresh-Liste ändert.

Die Funktionen LslRefreshListGetVarInfo(), LslRefreshListGetData() und LslRefreshListAdd() benötigen keine Online-Verbindung zur SPS, sondern nur die Verbindung zur Refresh-Liste die von der Funktion LslRefreshListCreateExt() hergestellt wird.

Weitere Informationen zur Callback-Funktion siehe Prototyp der Callbackfunktion im Kapitel LslRefreshListSetCallback.

Daten lesen ohne Eintrag in der Refresh-Liste

1. LslRefreshListGetVarInfo() ermittelt die Adresse des numerischen Servers (Integer oder REAL) oder des String-Servers oder der globalen Variable (Integer, REAL, LREAL, ARRAY und STRUCT).
2. Mit LslRefreshListGetData() oder LslRefreshListGetDataSize() können dann die Daten gelesen werden.

Daten schreiben ohne Eintrag in der Refresh-Liste

1. LslRefreshListGetVarInfo() ermittelt die Adresse des numerischen Servers (Integer oder REAL) oder der globalen Variable (Integer, REAL, LREAL, ARRAY und STRUCT).
2. Mit LslRefreshListSetData() können dann die Daten geschrieben werden.

Beispiel zur Refresh-Liste

```
#include "Lasal32\lsl_stdhdr.h"
#include "Lasal32\lsl_stdint.h"
#include "Lasal32\Lasal32.h"

#include <conio.h>
#include <string>
#include <thread>

// Wie müssen die Werte von der SPS gelesen werden
typedef enum
{
    RFRSHLIST_READTYPE_NUMSERVER,      //numerischer Server (Integer, REAL)
    RFRSHLIST_READTYPE_STRSERVER,      //String-Server
```

```

RFRSHLIST_READTYPE_VAR,           //Alle Variablen mit 1, 2 oder 4 Byte Länge
RFRSHLIST_READTYPE_ARRAY,         //Arrays
RFRSHLIST_READTYPE_STRUCT,        //Structs
RFRSHLIST_READTYPE_LREAL,         //LREAL
} RFRSHLIST_READ_TYPE;

//Wie müssen die Werte interpretiert werden
typedef enum
{
    RFRSHLIST_TYPE_STRING_A,      //ASCII-String
    RFRSHLIST_TYPE_SINT,          //Variable 1 Byte signed int
    RFRSHLIST_TYPE_INT,           //Variable 2 Byte signed int
    RFRSHLIST_TYPE_DINT,          //Variable 4 Byte signed int
    RFRSHLIST_TYPE_UINT,          //Variable 1 Byte unsigned int
    RFRSHLIST_TYPE_UINT,          //Variable 2 Byte unsigned int
    RFRSHLIST_TYPE_UDINT,         //Variable 4 Byte unsigned int
    RFRSHLIST_TYPE_REAL,          //Variable 4 Byte float
    RFRSHLIST_TYPE_LREAL,          //Variable 4 Byte double
    RFRSHLIST_TYPE_ARRAY_1,        //Array Typ1
    RFRSHLIST_TYPE_ARRAY_2,        //Array Typ2
    RFRSHLIST_TYPE_STRUCT_1,       //Struktur Typ1
    RFRSHLIST_TYPE_STRUCT_2,       //Struktur Typ2
} RFRSHLIST_DATA_TYPE;

//Index in MyVarInfo[]
#define VAR_ID_objRL_ServerData      0 //Numerical server
#define VAR_ID_StringTest_Data       1 //String server
#define VAR_ID_DataClass1_AddrStruct1 2 //Adress-Server für DataClass1.StructVar1
#define VAR_ID_DataClass1_StructVar1 3
#define VAR_ID_DataClass1_AddrStruct2 4 //Adress-Server für DataClass1.StructVar2
#define VAR_ID_DataClass1_StructVar2 5
#define VAR_ID_DataClass1_Array1     6 //Adress-Server für DataClass1.Array1
#define VAR_ID_DataClass1_Array1     7
#define VAR_ID_DataClass1_AddrArray2 8 //Adress-Server für DataClass1.Array2
#define VAR_ID_DataClass1_Array2     9

//Aufbau eines MyVarInfo[]-Elements
typedef struct {
    long                  iRLB;
    SRLVarInfo            VarInfo;
    RFRSHLIST_READ_TYPE   RdType;      //Wie müssen die Werte von der SPS gelesen werden
    RFRSHLIST_DATA_TYPE   DtType;      //Wie müssen die Werte interpretiert werden
    long                  size;        //Variablenlänge in Byte
    void*                 pDest;       //Ptr auf den Speicherort
} VarInfoStruct;

//typedefs which represent the structs used on the PLC
typedef struct {
    uint32_t uLongVar1;
    double LrealVar1;
    uint8_t CharArray1[32];
} StructType1;

typedef struct {

```

```

long iValue1;
long iValue2;
long iValue3;
} StructType2;

//*****
//Thread zum Lesen und auswerten einer Structure
//*****
void ReadStruct(VarInfoStruct* pVarInfo, uint32_t dwAddr)
{
    StructType1* pStructVar1;
    StructType2* pStructVar2;

    if (LslRefreshListGetData(pVarInfo->iRLB, &pVarInfo->VarInfo, (uint8_t*)pVarInfo->pDest, pVa
    {
        switch (pVarInfo->DtType)
    {
        case RFRSHLIST_TYPE_STRUCT_1:
    {
        pStructVar1 = (StructType1*)pVarInfo->pDest;
        printf("StructVar1.uLongVar1: %u\n", pStructVar1->uLongVar1);
        printf("StructVar1.LrealVar1: %g\n", pStructVar1->LrealVar1);
        printf("StructVar1.CharArray1: %s\n", pStructVar1->CharArray1);
    }
    break;
    case RFRSHLIST_TYPE_STRUCT_2:
    {
        pStructVar2 = (StructType2*)pVarInfo->pDest;
        printf("StructVar2: %d, %d, %d\n", pStructVar2->iValue1, pStructVar2->iValue2, pStruct
        }
    break;
    }
    else
        printf("LslRefreshListGetData() failed(0x%x)\n", GetLastError());
    }
}

//*****
//Thread zum Lesen und auswerten eines String-Servers
//*****
void ReadString(VarInfoStruct* pVarInfo, uint32_t dwAddr)
{
    uint32_t iTrueLenght = 0;

    if (LslRefreshListGetDataSize(pVarInfo->iRLB, &pVarInfo->VarInfo, &iTrueLenght, (uint8_t*)pVa
        printf("String server String=%s\n", (char*)pVarInfo->pDest);
    else
        printf("LslRefreshListGetDataSize() failed(0x%x)\n", GetLastError());
}

//*****
//Thread zum Lesen und auswerten eines Arrays
//*****
void ReadArray(VarInfoStruct* pVarInfo, uint32_t dwAddr)

```

```
{  
    long*          pArrayVar1;  
    unsigned long* pArrayVar2;  
  
    if (LslRefreshListGetData(pVarInfo->iRLB, &pVarInfo->VarInfo, (uint8_t*)pVarInfo->pDest, p  
    {  
        switch (pVarInfo->DtType)  
        {  
            case RFRSHLIST_TYPE_ARRAY_1:  
            {  
                pArrayVar1 = (long*)pVarInfo->pDest;  
                printf("Array1: %d, %d, %d\n", *pArrayVar1, *(pArrayVar1+1), *(pArrayVar1+2));  
            }  
            break;  
            case RFRSHLIST_TYPE_ARRAY_2:  
            {  
                pArrayVar2 = (unsigned long*)pVarInfo->pDest;  
                printf("Array2: %u, %u, %u\n", *pArrayVar2, *(pArrayVar2+1), *(pArrayVar2+2));  
            }  
            break;  
        }  
    }  
    else  
        printf("LslRefreshListGetData() failed(0x%x)\n", GetLastError());  
}  
  
//*****  
//Callbackfunction of the refreshlist. It gets called when a variable received a change.  
//*****  
void CallbackFunct(void* pCallbackData, uint32_t dwAddr, uint32_t dwVarID, int32_t nData)  
{  
    VarInfoStruct* pVarInfo = NULL;  
  
    pVarInfo = (VarInfoStruct*)pCallbackData;           //Ptr auf MyVarInfo[0]  
    pVarInfo += dwVarID;                                //dwVarID ist Index in MyVarInfo[]  
  
    switch (pVarInfo->RdType)  
    {  
        case RFRSHLIST_READTYPE_NUMSERVER:  
        {  
            //Just use the nData value  
            printf("Numerical server Value=%d\n", nData);  
        }  
        break;  
  
        case RFRSHLIST_READTYPE_STRUCT:  
        {  
            //Read the struct in a thread (LslRefreshListGetData() must not be called here)  
            std::thread th(ReadStruct, pVarInfo, dwAddr);  
            th.detach();  
        }  
        break;  
  
        case RFRSHLIST_READTYPE_STRSERVER:  
        {  
    }
```

```

//Read the string in a thread (LslRefreshListGetDataSize() must not be called here)
std::thread th(ReadString, pVarInfo, dwAddr);
    th.detach();
}
break;

case RFRSHLIST_READTYPE_ARRAY:
{
    //Read the array in a thread (LslRefreshListGetData() must not be called here)
    std::thread th(ReadArray, pVarInfo, dwAddr);
    th.detach();
}
break;
}

//*****
int main()
//*****
{
    static VarInfoStruct MyVarInfo[10];           //Var infos, ein Element für jede Variable
    VarInfoStruct* pVarInfoAddr;
    VarInfoStruct* pVarInfo;

    //Speicher für Server, String-Server und Member-Variable (struct und array)
    static long          objRL_ServerData;
    static char          StringTest_Data[256];
    static StructType1   DataClass1_StructVar1;
    static StructType2   DataClass1_StructVar2;
    static char          DataClass1_Array1[2048];
    static unsigned long DataClass1_Array2[3];

    int iRefreshList = 0;
    int iOcbNr = 0;

    std::string strOnlineConn = "TCP:10.10.170.190";
    std::string strRefreshConn = strOnlineConn + ";ApplID=10";

    //Create the refreshlist
    iRefreshList = LslRefreshListCreateExt(strOnlineConn.c_str(), 0, 0, 0, NULL, CallbackFunct, 1);
    if (iRefreshList)
    {
        //Register a normal server at the refreshlist
        pVarInfo = &MyVarInfo[VAR_ID_objRL_ServerData];
        if (LslRefreshListGetVarInfo(iRefreshList, "objRL.ServerData", &pVarInfo->VarInfo))
        {
            if (LslRefreshListAdd(iRefreshList, &pVarInfo->VarInfo, VAR_ID_objRL_ServerData,
            {
                pVarInfo->iRLB = iRefreshList;
                pVarInfo->RdType = RFRSHLIST_READTYPE_NUMSERVER;
                pVarInfo->DtType = RFRSHLIST_TYPE_DINT;
                pVarInfo->size = sizeof(objRL_ServerData); //sizeof destination
                pVarInfo->pDest = &objRL_ServerData;
            }
        }
    }
}

```

```
        }

    //Add a String server to the refreshlist
    pVarInfo = &MyVarInfo[VAR_ID_StringTest_Data];
    if (LslRefreshListGetVarInfo(iRefreshList, "StringTest.Data", &pVarInfo->VarInfo
    {
        if (LslRefreshListAdd(iRefreshList, &pVarInfo->VarInfo, VAR_ID_StringTest_Data
        {
            pVarInfo->iRLB = iRefreshList;
            pVarInfo->RdType = RFRSHLIST_READTYPE_STRSERVER;
            pVarInfo->DtType = RFRSHLIST_TYPE_STRING_A;
            pVarInfo->size = sizeof(StringTest_Data); //sizeof destination
            pVarInfo->pDest = StringTest_Data;
        }
    }

    //Add the member variable <DataClass1.StructVar1> to the refreshlist
    //Get the address of the server "AddrStruct1"
    pVarInfoAddr = &MyVarInfo[VAR_ID_DataClass1_AddrStruct1];
    if (LslRefreshListGetVarInfo(iRefreshList, "DataClass1.AddrStruct1", &pVarInfoAddr
    {
        //Read the address of the structure from the server "AddrStruct1"
        pVarInfo = &MyVarInfo[VAR_ID_DataClass1_StructVar1];
        if (LslRefreshListGetData(iRefreshList, &pVarInfoAddr->VarInfo, (uint8_t*)&pVa
        {
            //Use its value as address and add the struct to the refreshlist
            pVarInfo->VarInfo.dwSize = sizeof(DataClass1_StructVar1);
            pVarInfo->VarInfo.dwType = LSL_RL_TYPE_GLOBAL;
            if (LslRefreshListAdd(iRefreshList, &pVarInfo->VarInfo, VAR_ID_DataClass1_St
            {
                pVarInfo->iRLB = iRefreshList;
                pVarInfo->RdType = RFRSHLIST_READTYPE_STRUCT;
                pVarInfo->DtType = RFRSHLIST_TYPE_STRUCT_1;
                pVarInfo->size = sizeof(DataClass1_StructVar1); //sizeof destination
                pVarInfo->pDest = &DataClass1_StructVar1;
            }
        }
    }

    //Add the member variable <DataClass1.StructVar2> to the refreshlist
    //Get the address of the server "AddrStruct2"
    pVarInfoAddr = &MyVarInfo[VAR_ID_DataClass1_AddrStruct2];
    if (LslRefreshListGetVarInfo(iRefreshList, "DataClass1.AddrStruct2", &pVarInfoAddr
    {
        //Read the address of the structure from the server "AddrStruct2"
        pVarInfo = &MyVarInfo[VAR_ID_DataClass1_StructVar2];
        if (LslRefreshListGetData(iRefreshList, &pVarInfoAddr->VarInfo, (uint8_t*)&pVa
        {
            //Use its value as address and add the struct to the refreshlist
            pVarInfo->VarInfo.dwSize = sizeof(DataClass1_StructVar2);
            pVarInfo->VarInfo.dwType = LSL_RL_TYPE_GLOBAL;
            if (LslRefreshListAdd(iRefreshList, &pVarInfo->VarInfo, VAR_ID_DataClass1_St
            {
                pVarInfo->iRLB = iRefreshList;
```

```

pVarInfo->RdType = RFRSHLIST_READTYPE_STRUCT;
pVarInfo->DtType = RFRSHLIST_TYPE_STRUCT_2;
pVarInfo->size = sizeof(DataClass1_StructVar2);      //sizeof destination
pVarInfo->pDest = &DataClass1_StructVar2;
}
}
}

//Add the member variable <DataClass1.Array1> to the refreshlist
//Get the address of the server "AddrArray1"
pVarInfoAddr = &MyVarInfo[VAR_ID_DataClass1_AddrArray1];
if (LslRefreshListGetVarInfo(iRefreshList, "DataClass1.AddrArray1", &pVarInfoAddr)
{
    //Read the address of the array from the server "AddrArray1"
    pVarInfo = &MyVarInfo[VAR_ID_DataClass1_Array1];
    if (LslRefreshListGetData(iRefreshList, &pVarInfoAddr->VarInfo, (uint8_t*)&pVarI
    {
        //Use its value as address and add the array to the refreshlist
        pVarInfo->VarInfo.dwSize = sizeof(DataClass1_Array1);
        pVarInfo->VarInfo.dwType = LSL_RL_TYPE_GLOBAL;
        if (LslRefreshListAdd(iRefreshList, &pVarInfo->VarInfo, VAR_ID_DataClass1_Arra
        {
            pVarInfo->iRLB = iRefreshList;
            pVarInfo->RdType = RFRSHLIST_READTYPE_ARRAY;
            pVarInfo->DtType = RFRSHLIST_TYPE_ARRAY_1;
            pVarInfo->size = sizeof(DataClass1_Array1);          //sizeof destination
            pVarInfo->pDest = &DataClass1_Array1;
        }
    }
}

//Add the member variable <DataClass1.Array2> to the refreshlist
//Get the address of the server "AddrArray2"
pVarInfoAddr = &MyVarInfo[VAR_ID_DataClass1_AddrArray2];
if (LslRefreshListGetVarInfo(iRefreshList, "DataClass1.AddrArray2", &pVarInfoAddr)
{
    //Read the address of the array from the server "AddrArray2"
    VarInfo = &MyVarInfo[VAR_ID_DataClass1_Array2];
    if (LslRefreshListGetData(iRefreshList, &pVarInfoAddr->VarInfo, (uint8_t*)&pVarI
    {
        //Use its value as address and add the array to the refreshlist
        pVarInfo->VarInfo.dwSize = sizeof(DataClass1_Array2);
        pVarInfo->VarInfo.dwType = LSL_RL_TYPE_GLOBAL;
        if (LslRefreshListAdd(iRefreshList, &pVarInfo->VarInfo, VAR_ID_DataClass1_Arra
        {
            pVarInfo->iRLB = iRefreshList;
            pVarInfo->RdType = RFRSHLIST_READTYPE_ARRAY;
            pVarInfo->DtType = RFRSHLIST_TYPE_ARRAY_2;
            pVarInfo->size = sizeof(DataClass1_Array2);          //sizeof destination
            pVarInfo->pDest = &DataClass1_Array2;
        }
    }
}

//Start the refreshlist

```

```
LslRefreshListStart(iRefreshList, LSL_RL_FLAG_PERMANENT_LIST); // start refreshlist
_getch();

//Write data to the struct
StructType1 structData1;
StructType2 structData2;

structData1.uLongVar1 = 0;
structData1.LrealVar1 = 0.0;
structData1.CharArray1[0] = 0;

structData2.iValue1 = 0;
structData2.iValue2 = 0;
structData2.iValue3 = 0;

//Create an Online connection
iOcbNr = OcbOpen();
bool bOnline = OnlineH(iOcbNr, strOnlineConn.c_str(), 0, 0, 0, 0) == TRUE;
if (bOnline)
{
    //Two differnt ways to write to a struct
    //Writing using refreshlist
    LslRefreshListSetData(iRefreshList, &MyVarInfo[VAR_ID_DataClass1_StructVar1].VarInfo,
    //Writing using normal connection
    LslSetDataExH(iOcbNr, (uint8_t*)&structData2, MyVarInfo[VAR_ID_DataClass1_StructVar1].dwAddr, dwnewValue);

    //Write data to a numerical server
    DWORD dwnewValue = 0;
    //Two differnt ways to write to a server
    //Writing using refreshlist
    LslRefreshListSetData(iRefreshList, &MyVarInfo[VAR_ID_objRL_ServerData].VarInfo,
    //Writing using normal connection
    dwnewValue = -99999999;
    LslWriteToSvrH(iOcbNr, MyVarInfo[VAR_ID_objRL_ServerData].VarInfo.dwAddr, dwnewValue);

    //Write data to a string server
    std::string strnewValue = "Test";
    //Two differnt ways to write to a string server
    //Writing using refreshlist
    LslRefreshListSetData(iRefreshList, &MyVarInfo[VAR_ID_StringTest_Data].VarInfo, (void*)&strnewValue,
    //Writing using normal connection
    std::string strnewValue1 = "Test_Test_Test";
    LslWriteToSvrStrH(iOcbNr, MyVarInfo[VAR_ID_StringTest_Data].VarInfo.dwAddr, strnewValue1);
}

_getch();
//Destroy the refresh list at the end
LslRefreshListDestroy(iRefreshList);

//Release the normal connection
OfflineH(iOcbNr);
OcbClose(iOcbNr);
}
```

6.1.12.1 LslRefreshListAdd

Diese Funktion fügt einen neuen Eintrag am Ende der Refresh-Liste ein. Die Information über die Variable kann unter anderem mit der Funktion [LslRefreshListGetVarInfo\(\)](#) ermittelt werden. Bei der Wahl der Refresh-Zeit (Übergabeparameter dwRefreshTime) sollten Überlegungen angestellt werden, wie oft sich dieser Wert ändert bzw. in welchem Zeitraster dieser von der SPS überwacht werden soll. Da das menschliche Auge sehr träge ist und jedes Display nachleuchtet, ist bei einem am Bildschirm dargestellten Wert eine Refreshrate von unter 50 ms als unangenehm zu betrachten.

Faustregel

250 ms	„Langsame“ Werte
100 ms	„Standard“ Werte
50 ms	„Schnelle“ Werte

Es ist darauf zu achten, welche der beiden Refresh-Listen mit dem Übergabeparameter dwFlag angesprochen wird.



extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListAdd(int iRLB, const SRLVarInfo *pVarInfo, DW...

Übergabeparameter	Typ	Beschreibung
iRLB	INT	ID der Refresh-Liste
pVarInfo	const SRLVarInfo	Zeiger auf SRLVarInfo (z.B. erstellt durch LslRefreshListGetVarInfo())
dwVarID	DWORD	Anwender-spezifische ID zur eindeutigen Identifikation dieses Eintrages
dwRefreshTime	DWORD	Refresh-Zeit dieser Variable; die minimale Zeit beträgt 10 ms
dwFlag	DWORD	LSL_RL_FLAG_DYNAMIC_LIST oder LSL_RL_FLAG_PERMANENT_LIST oder LSL_RL_FLAGADD_NOANSWER (wartet nicht auf eine Antwort, nachdem eine neue Variable hinzugefügt wurde. Es ist schneller, aber prüft nicht, ob die Variable richtig hinzufügt wurde)
Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich FALSE Fehler

		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.
--	--	---------------------------------------------------------------------------------------------------------

6.1.12.2 LslRefreshListClear

Diese Funktion löscht alle Einträge aus der Refresh-Liste. Anschließend ist die RefreshList-ID selbstverständlich weiter gültig und die Callback-Funktion ebenfalls noch eingetragen.



Es ist darauf zu achten, welche der beiden Refresh-Listen angesprochen wird.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListClear(int iRLB, DWORD dwFlag);
```

Übergabeparameter	Typ	Beschreibung	
iRLB	INT	ID der Refresh-Liste	
dwFlag	DWORD	LSL_RL_FLAG_DYNAMIC_LIST oder LSL_RL_FLAG_PERMANENT_LIST	
		Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	

6.1.12.3 LslRefreshListCreate

Diese Funktion erstellt eine neue Refresh-Liste. Im Fehlerfall wird der Wert 0 retourniert, ansonsten eine gültige ID. Diese ID ist bei jedem programmtechnischen Zugriff auf die Refresh-Liste notwendig.

```
extern "C" int LASAL32_EXPORTS LslRefreshListCreate();
```

Übergabeparameter	Typ	Beschreibung	
keine			

Rückgabeparameter	Typ	Beschreibung
		0 Fehler ≠0 ID der Refresh-Liste

6.1.12.4 LslRefreshListCreateExt

Legt mehrerer Funktionen zusammen ([LslRefreshListCreate\(\)](#), [LslRefreshListOnline\(\)](#), [LslRefresListSetCallback\(\)](#), [LslRefreshListSymbolTableInit\(\)](#)).

```
extern "C" int LslRefreshListCreateExt(
  const char          *szComm,
  uint8_t              uBaudRate,
  uint8_t              uPcStation,
  uint8_t              uSpsStation,
  const char          *lpcName,
  CB_RLADD_FUNCTYPE  *pCallback,
  void                *pCallbackData,
  uint32_t             dwTimeoutMS
);
```

Übergabeparameter	Typ	Beschreibung
szComm	const char*	ASCII-0-String, welcher die verwendete Schnittstelle zum Aufbau der Verbindung spezifiziert Beispiel: TCP:10.10.116.42
uBaudRate	uint8_t	0; derzeit nicht unterstützt
uPcStation	uint8_t	0; derzeit nicht unterstützt
uSpsStation	uint8_t	0; derzeit nicht unterstützt
lpcName	const char*	Objektname der Symboltabelle
pCallback	CB_RLADD_FUNCTYPE *	Zeiger auf die aktuelle Callback-Funktion. Der Prototyp dieser Funktion ist CB_RLADD_FUNCTYPE (siehe unten).
pCallbackData	void*	Void-Zeiger, der bei jedem Aufruf der Callback-Funktion übergeben wird.
dwTimeoutMS	DWORD	Wenn der angegebene Wert ungleich 0 ist, dann überprüft das Kommando vor der Erstellung der Refresh-Liste, ob in der angegeben Zeit online gegangen werden kann und ob sich die Steuerung in RunRAM oder RunROM befindet.
Rückgabeparameter	Typ	Beschreibung

	INT	0 Fehler ≠0 ID der angelegten Refresh-Liste
--	-----	------------------------------------------------

Beispiel

Siehe Verwalten der Refresh-Liste.

6.1.12.5 LslRefreshListDestroy

Diese Funktion löscht eine bereits erstellte Refresh-Liste. Als Übergabeparameter ist die ID der zu löschenenden Refresh-Liste notwendig. Normalerweise wird 0 zurückgegeben. Im Fehlerfall ist der Rückgabewert ungleich 0. Eine eventuell bestehende Verbindung zur SPS wird selbstverständlich gelöscht. Anschließend darf die RefreshList-ID nicht mehr verwendet werden.

Diese Funktion liefert 0 und ≠ 0 als Rückgabewert. So manch andere Funktionen liefern TRUE bzw. FALSE.



```
extern "C" int LASAL32_EXPORTS LslRefreshListDestroy(int iRLB);
```

Übergabeparameter	Typ	Beschreibung	
iRLB	INT	ID der Refresh-Liste	
Rückgabeparameter	Typ	Beschreibung	
		0 Funktion erfolgreich ≠0 Fehler	

Mit [GetLastError\(\)](#) kann der letzte Fehler-Code ermittelt werden. Dieser Code entspricht entweder einem Lasal32 Fehler-Code oder einem systemspezifischen Fehler (Windows: [GetLastError\(\)](#), oder Linux: errno).

6.1.12.6 LslRefreshListGetData

Mit dieser Funktion kann der Wert einer Variable oder eines Server ohne einem Eintrag in RefreshList gelesen werden.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListGetData(int iRLB, const SRLVarInfo *pVarInfo
```

Übergabeparameter		Typ	Beschreibung
iRLB		INT	ID der Refresh-Liste
pVarInfo		CONST SRLVarInfo	Zeiger auf bereits initialisierte Struktur SRLVarInfo
pData		BYTE	Zeiger auf einen Byte-Puffer (Destination)
dwSize		DWORD	Größe des Byte-Puffers in Byte
Rückgabeparameter		Typ	Beschreibung
			<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>

Beispiel 1 (String-Server lesen)

```
int iRLB = LslRefreshListCreate();
if (iRLB > 0)
{
    LslRefreshListOnline(iRLB, "TCP:10.10.170.190", 0, 0, 0);

    if (LslRefreshListIsOnline(iRLB))
    {
        SRLVarInfo varInfo;

        if (LslRefreshListGetVarInfo(iRLB, "StringR1.Data", &varInfo) == TRUE)
        {
            printf("LslRefreshListGetVarInfo: '%s'\n", "StringR1.Data");
            printf("  Address : 0x%08X, %u\n", varInfo.dwAddr, varInfo.dwAddr);
            printf("  Type    : 0x%08X, %u\n", varInfo.dwType, varInfo.dwType);
            printf("  Size    : %u\n", varInfo.dwSize);

            uint8_t byaBuf[2000];
            if (LslRefreshListGetData(iRLB, &varInfo, byaBuf, sizeof(byaBuf)) == FALSE)
                printf("Failed reading string server (0x%08X)!\n", GetLastError());
            else
                printf("Ok (%s)\n", byaBuf);
        }
        else
            printf("LslRefreshListGetVarInfo Error=0x%08X\n", GetLastError());
    }
}
```

```
}
```

```
LslRefreshListDestroy(iRLB);
```

Beispiel 2 (numerischen Server lesen)

```
int iRLB = LslRefreshListCreate();
if (iRLB > 0)
{
    LslRefreshListOnline(iRLB, "TCP:10.10.170.190", 0, 0, 0);

    if (LslRefreshListIsOnline(iRLB))
    {
        SRLVarInfo varInfo;

        if (LslRefreshListGetVarInfo(iRLB, "objRL.ServerCmd", &varInfo) == TRUE)
        {
            printf("LslRefreshListGetVarInfo: '%s'\n", "objRL.ServerCmd");
            printf("  Address : 0x%08X, %u\n", varInfo.dwAddr, varInfo.dwAddr);
            printf("  Type    : 0x%08X, %u\n", varInfo.dwType, varInfo.dwType);
            printf("  Size    : %u\n", varInfo.dwSize);

            int32_t intVar;
            if (LslRefreshListGetData(iRLB, &varInfo, (uint8_t*)&intVar, sizeof(intVar)) == FALSE)
                printf("Failed reading server (0x%08X)!\n", GetLastError());
            else
                printf("Ok (%d)\n", intVar);
        }
        else
            printf("LslRefreshListGetVarInfo Error=0x%08X\n", GetLastError());
    }
}
LslRefreshListDestroy(iRLB);
```

Beispiel 3 (globale Variable lesen)

```
int iRLB = LslRefreshListCreate();
if (iRLB > 0)
{
    LslRefreshListOnline(iRLB, "TCP:10.10.170.190", 0, 0, 0);

    if (LslRefreshListIsOnline(iRLB))
    {
        SRLVarInfo varInfo;

        if (LslRefreshListGetVarInfo(iRLB, "g_BigBuf", &varInfo) == TRUE)
        {
            printf("LslRefreshListGetVarInfo: '%s'\n", "g_BigBuf");
            printf("  Address : 0x%08X, %u\n", varInfo.dwAddr, varInfo.dwAddr);
            printf("  Type    : 0x%08X, %u\n", varInfo.dwType, varInfo.dwType);
            printf("  Size    : %u\n", varInfo.dwSize);

            uint8_t byaBuf[2000];
            if (LslRefreshListGetData(iRLB, &varInfo, byaBuf, sizeof(byaBuf)) == FALSE)
                printf("Failed reading global variable (0x%08X)!\n", GetLastError());
        }
    }
}
```

```

        else
        {
            byaBuf[1999] = 0;
            printf("Ok (%s)\n", byaBuf);
        }
    }
    else
        printf("LslRefreshListGetVarInfo Error=0x%08X\n", GetLastError());
}
}
LslRefreshListDestroy(iRLB);

```

6.1.12.7 LslRefreshListGetDataSize

Mit dieser Funktion können der Wert und die Länge einer Variable oder eines Servers gelesen werden. Dazu ist kein Eintrag in der Refresh-Liste nötig. Bei einem String-Server wird die richtige String-Länge in pSize zurück geliefert und im Puffer pData der 0-terminierte String, wenn der Puffer groß genug ist.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListGetDataSize(
    int32_t           iRLB,
    const SRLVarInfo* pVarInfo,
    DWORD*            pSize,
    unsigned char*    pData,
    DWORD             dwSize
);

```

Übergabeparameter	Typ	Beschreibung
iRLB	int32_t	ID der Refresh-Liste (Ergebnis von LslRefreshListCreate() oder LslRefreshListCreateExt())
pVarInfo	const SRLVarInfo*	Zeiger auf SRLVarInfo (erstellt durch LslRefreshListGetVarInfo())
pSize	DWORD*	Zeiger auf eine Variable, wo bei einem String-Server die String-Länge ohne 0-Terminator eingetragen wird
pData	unsigned char*	Zeiger auf den Ergebnispuffer
dwSize	DWORD	Größe von pData in Byte. Muss für den 0-Terminator um eins größer als die in pSize angezeigte String-Länge sein.
Rückgabeparameter	Typ	Beschreibung
		<div style="display: flex; justify-content: space-between;"> <div>TRUE</div> <div>Funktion erfolgreich</div> </div> <div style="display: flex; justify-content: space-between;"> <div>FALSE</div> <div>Fehler</div> </div>
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.

Beispiel

Damit können sie wie folgt optimiert einen String lesen. Dieser Ablauf hat den Vorteil, dass man bei kleineren Strings nicht noch einmal eine Netzwerkkommunikation machen muss, da der String bereits gelesen wurde, und in buf enthalten ist. Nur wenn dieser zu klein ist, muss man noch einmal mit einem größeren Puffer lesen.

```
static int read_string_server(int32_t iRLB, const SRLVarInfo* pVarInfo, std::string& str)
{
#define BUF_SIZE 255
    uint8_t buf[BUF_SIZE + 1];
    uint32_t size_string;

    if (LslRefreshListGetDataSize(iRLB, pVarInfo, &size_string, buf, BUF_SIZE)) {
        if (size_string > BUF_SIZE) {
            // my local buffer is too small
            uint8_t* ptr = (uint8_t*)malloc(size_string + 1);
            if (!ptr)
                return -ENOMEM;
            if (LslRefreshListGetDataSize(iRLB, pVarInfo, NULL, ptr, size_string + 1))
            {
                str = (char*)ptr;
                free(ptr);
            }
        } else {
            // LslRefreshListGetDataSize failed
            free(ptr);
            return -EOTHER;
        }
    } else {
        // buf is null terminated
        str = (char*)buf;
    }
    return 0; // 0 mean success
}
// LslRefreshListGetDataSize failed
return -EOTHER;
}
```

6.1.12.8 LslRefreshListGetLoaderVersion

Gibt die Versionsnummer des Loaders zurück.

```
extern "C" DWORD LASAL32_EXPORTS LslRefreshListGetLoaderVersion(
    int32_t    iRLB
);
```

Übergabeparameter	Typ	Beschreibung
-------------------	-----	--------------

iRLB	int32_t	ID der Refresh-Liste (Ergebnis von LslRefreshListCreate() oder LslRefreshListCreateExt())				
Rückgabeparameter	Typ	Beschreibung				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">>0</td><td>Loader-Versionsnummer</td></tr> <tr> <td>0</td><td>Fehler</td></tr> </table>	>0	Loader-Versionsnummer	0	Fehler
>0	Loader-Versionsnummer					
0	Fehler					
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.						

Die Loader-Versionsnummer hat folgenden Aufbau:

HIGH-WORD				LOW-WORD			
31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
				Major	Minor	Subversion	

Beispiel

Die Loader-Version V02.02.85 entspricht dem Hex-Wert 16#00002255.

6.1.12.9 LslRefreshListGetVarInfo

Diese Funktion ermittelt die notwendigen Informationen über die angegebene Variable. Zurzeit werden nur Server mit der Schreibweise Objectname.Servername sowie globale Variablen (einfache Variable wie INT, SINT, DINT, UINT, USINT, UDINT, REAL, LREAL sowie Strukturen und Arrays) unterstützt. Für alle globalen Variablen werden nur die Adresse und der Typ ermittelt, aber keine Länge. Die Länge muss vom Aufrufer nachträglich in die SRLVarInfo-Struktur eingetragen werden.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListGetVarInfo(int iRLB, const char *lpcName, SR...
```

Übergabeparameter		Type	Beschreibung
iRLB		INT	ID der Refresh-Liste
lpcName		CONST CHAR	Name des Servers oder der Variable (ASCII-0-String)
pVarInfo		SRLVarInfo	Zeiger auf SRLVarInfo; diese Struktur wird von dieser Funktion initialisiert
Rückgabeparameter	Typ	Beschreibung	
		TRUE Funktion erfolgreich	

		FALSE Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.		

6.1.12.10 LslRefreshListIsOnline

Diese Funktion prüft eine bereits bestehende Online-Verbindung. Im Falle einer gültigen Verbindung wird TRUE retourniert. Besteht keine gültige Verbindung mehr, so wird FALSE zurückgegeben.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListIsOnline(int iRLB);
```

Übergabeparameter	Typ	Beschreibung	
iRLB	INT	ID der Refresh-Liste	
Rückgabeparameter	Typ	Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.			

6.1.12.11 LslRefreshListOffline

Diese Funktion beendet die Verbindung mit der SPS und stoppt somit den Datenaustausch zwischen SPS und der externen Station. Es werden außerdem alle Einträge aus der Refresh-Liste entfernt, die eigentliche Refresh-Liste jedoch nicht gelöscht und der Zeiger auf die Callback-Funktion bleibt erhalten. Die Refreshlist-ID behält somit ihre Gültigkeit und kann weiter verwendet werden.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListOffline(int iRLB);
```

Übergabeparameter	Typ	Beschreibung	
iRLB	INT	ID der Refresh-Liste	
Rückgabeparameter	Typ	Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.			

		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.
--	--	---------------------------------------------------------------------------------------------------------

6.1.12.12 LslRefreshListOnline

Diese Funktion wird zum Verbindungsaufbau zur SPS verwendet. Als Übergabeparameter sind die Refreslist-ID und die Verbindungsdaten notwendig.

Zur Zeit wird nur die TCP-Verbindung unterstützt.



extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListOnline(int iRLB, const char* szComm, BYTE uBaudRate, BYTE uPcStation, BYTE uSpStation)

Übergabeparameter		Typ	Beschreibung
iRLB	INT	ID der Refresh-Liste	
szComm	CONST CHAR	ASCII-0-String, welcher die verwendete Schnittstelle zum Aufbau der Verbindung spezifiziert. Beispiel: TCP:10.10.116.42	
uBaudRate	BYTE	0; derzeit nicht unterstützt	
uPcStation	BYTE	0; derzeit nicht unterstützt	
uSpStation	BYTE	0; derzeit nicht unterstützt	
Rückgabeparameter		Typ	Beschreibung
	LSL_BOOL		<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>

6.1.12.13 LslRefreshListSetCallback

Mit dieser Funktion wird die Callback-Funktion der Refresh-Liste initialisiert. Die Callback-Funktion dient dem eigentlichen Datentransfer, sie wird bei jeder Änderung der in die Refresh-Liste eingetragenen Daten aufgerufen.

extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListSetCallback(int iRLB, CB_RLADD_FUNCTYPE *pCa...

Übergabeparameter		Typ	Beschreibung	
iRLB		INT	ID der Refresh-Liste	
pCallback		CB_RLADD_FUNCTYPE	Zeiger auf die eigentliche CallBack-Funktion. Der Prototyp dieser Funktion ist CB_RLADD_FUNCTYPE (siehe unten).	
pCallbackData		VOID	Ein void-Pointer welcher bei jedem Aufruf der Callback-Funktion übergeben wird.	
Rückgabeparameter		Typ	Beschreibung	
			TRUE	Funktion erfolgreich
			FALSE	Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.				

Prototyp der Callbackfunktion

```
void __cdecl RTRLCounterCallback(void *pCallbackData, DWORD dwAddr, DWORD dwVarID, int dNewData)
```

Übergabeparameter		Typ	Beschreibung	
pCallbackData		VOID	Anwender definierte void-Zeiger (siehe oben)	
dwAddr		DWORD	Vom System verwaltete eindeutige Adresse	
dwVarID		DWORD	Anwenderspezifische ID des Eintages in der RefreshList (siehe Funktion LsRefreshListAdd())	
dNewData		INT	Geänderter (neuer) Wert der SPS	

Der Aufruf der Callback-Funktion wird nach dem Start der Refresh-Liste vom System übernommen. Der Anwender muss nur mehr die geänderten Daten entsprechend übernehmen und ggf. weitere Aktionen, seine Applikation betreffend, starten. Nach dem Stop der Refresh-Liste wird diese Funktion nicht mehr aufgerufen.

Es ist darauf zu achten, dass die Callback-Funktion so schnell wie möglich abgearbeitet wird, d.h. der darin befindliche Programmcode sollte so kurz wie möglich sein.

Es sollten keine weiteren Aufrufe von Lasal32.dll-Funktionen im Callback enthalten sein.



```
typedef void __cdecl CB_RLADD_FUNCTYPE (
    void    *pCallbackData,
    DWORD   dwAddr,
```

```

    DWORD dwVarID,
    int    dNewData
);

```

Bemerkungen

1. Wenn sich der Wert eines numerischen Servers (Ganzzahl oder Gleitpunktzahl REAL) ändert, dann wird der neue Wert im Parameter dNewData an die Callback-Funktion übergeben.
2. Wenn sich der Wert eines String-Servers ändert, dann muss der neue Wert vom String-Server gelesen werden. Dazu eignet sich gut die Funktion LslRefreshListGetDataSize(). Es kann aber auch die Funktion LslRefreshListGetData() verwendet werden. Diese Funktionen müssen aus einem eigenen Thread aufgerufen werden.
3. Wenn sich der Wert einer globalen Variablen ändert, dann muss der neue Wert von der SPS gelesen werden. Dazu eignet sich gut die Funktion LslRefreshListGetData(). Es kann aber auch die Funktion LslRefreshListGetDataSize() verwendet werden. Diese Funktionen müssen aus einem eigenen Thread aufgerufen werden.
4. Objekt-Member-Variablen oder mit malloc() allokierte Speicherbereiche wurden ja als LSL_RL_TYPE_GLOBAL deklariert und sind daher so zu behandeln wie in Punkt 3. beschrieben.

Die Funktionen LslRefreshListGetData() und LslRefreshListGetDataSize() benötigen keine Online-Verbindung zur SPS, sondern nur die Online-Verbindung zur Refresh-Liste die von der Funktion LslRefreshListCreateExt() hergestellt wird.

6.1.12.14 LslRefreshListSetData

Diese Funktion schreibt Daten auf eine Variable oder einen Server. Es ist nicht relevant, ob dieser Server (oder diese Variable) auch in der Refresh-Liste eingetragen ist.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListSetData(int iRLB, const SRLVarInfo *pVarInfo
```

Übergabeparameter	Typ	Beschreibung
iRLB	INT	ID der Refresh-Liste
pVarInfo	CONST SRLVarInfo	Zeiger auf bereits initialisierte Struktur SRLVarInfo
pData	CONST BYTE	Adresse der Daten (Source)
dwSize	DWORD	Größe der Daten in Byte
Rückgabeparameter	Typ	Beschreibung

TRUE Funktion erfolgreich

FALSE Fehler

Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion [GetLastError\(\)](#) auf.

Beispiel (C/C++)

```
// Globaler Zähler
DWORD g_dwCounterSpeedTest = 0;

// Callback Funktion (erhöht den globalen Zähler)
void __cdecl RTRLCounterCallback(
    void *pCallbackData,
    DWORD dwAddr,
    DWORD dwVarID,
    int dNewData);

{
    g_dwCounterSpeedTest++;
}

// Test Funktion
void TestRefreshListSpeed(int iRLB)
{
    g_dwCounterSpeedTest = 0;
    DWORD dwStart;

    SRLVarInfo varInfoStart, varInfoCounter;

    // Ermitteln von Infos für die Variable g_RTRFStart
    if (LslRefreshListGetVarInfo(iRLB, "g_RTRFStart", &varInfoStart) == TRUE)
    {
        // Ermitteln von Infos für einen Server.
        if (LslRefreshListGetVarInfo(iRLB, "ClassRealTimeRefreshList1.SrvCounter", &varInfoCounter) == TRUE)
        {
            dwWait = 3000;

            printf("RealTime ReafreshList Counter Test is running !\n");

            // Setze Callback Funktion
            LslRefreshListSetCallback(iRLB, RTRLCounterCallback, NULL);
            // Füge Variable zur RefreshListe hinzu.
            LslRefreshListAdd(iRLB, &varInfoCounter, 1234, 100, 0);
            // Starte RefreshListe
            LslRefreshListStart(iRLB, 0);

            Sleep(10);

            dwStart = 1;
        }
    }
}
```

```
// Starte den internen Zähler in der Steuerung.
LslRefreshListSetData(iRLB, &varInfoStart, (const BYTE *) &dwStart,
sizeof(dwStart));

// Warte eine gewisse Zeit.
// In dieser Zeit wird der Callback aufgerufen.
Sleep(dwWait);

dwStart = 0;
// Stoppe den inneren Zähler in der Steuerung.
LslRefreshListSetData(iRLB, &varInfoStart, (const BYTE *) &dwStart,
sizeof(dwStart));

// Lösche die RefreshListe
LslRefreshListClear(iRLB, 0);
// Zeige Messergebnisse an.
printf("g_dwCounterSpeedTest=%u, one need=% .2f\n",
g_dwCounterSpeedTest,
double(dwWait) / double(g_dwCounterSpeedTest));
}
else
printf("Error: LslRefreshListGetVarInfo(ClassRealTimeRefreshList1.SrvCounter)\n");
}
else
printf("Error:LslRefreshListGetVarInfo(g_RTRFStart)\n";
}

void TestRefreshList()
{
// Erzeuge einen RefreshListBlock.
int iRLB = LslRefreshListCreate();

// Gehe per TCP auf der Steuerung Online
if (LslRefreshListOnline(iRLB, "TCP:10.10.116.43", 0, 0, 0))
{
// Prüfe, ob Online gehen funktioniert hat.
if (LslRefreshListIsOnline(iRLB))
printf("Online ok\n");
else
printf("Online NOT ok\n");

// Führe Test Funktion aus.
TestRefreshListSpeed(iRLB);

// Gehe wieder Offline
LslRefreshListOffline(iRLB);
}
else
printf("Online error 0x%08X\n", GetLastError());
}

// Lösche RefreshListBlock.
LslRefreshListDestroy(iRLB);
}
```

Für dieses Beispiel muss auf der Steuerung die passende Applikation laufen!

6.1.12.15 LslRefreshListSymbolTableInit

```
extern "C" uint32_t LslRefreshListSymbolTableInit(int iRLB, const char *lpcName);
```

Übergabeparameter	Typ	Beschreibung
iRLB	INT	ID der Refresh-Liste
lpcName	CONST CHAR	Objektname der Symboltabelle
Übergabeparameter	Typ	Beschreibung
		Im Fehlerfall eine RefreshList ID = 0, ansonsten wird ein Wert ungleich 0 zurückgeliefert

6.1.12.16 LslRefreshListStartCount

Mit dieser Funktion kann die Anzahl der Einträge in einer Refresh-Liste gesteuert werden. Bei dwCount soll natürlich nie mehr als die tatsächliche Anzahl der Einträge übergeben werden. Wird als Anzahl 0 übergeben, verhält sich diese Funktion wie [LslRefreshListClear\(\)](#).

Es ist darauf zu achten, welche der beiden Refresh-Listen angesprochen wird.



```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListStartCount(int iRLB, DWORD dwCount, DWORD dwFlag);
```

Übergabeparameter	Typ	Beschreibung
iRLB	INT	ID der Refresh-Liste
dwCount	DWORD	Anzahl der gültigen Einträge in RefreshList
dwFlag	DWORD	LSL_RL_FLAG_DYNAMIC_LIST oder LSL_RL_FLAG_PERMANENT_LIST
Übergabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich FALSE Fehler

		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.
--	--	---------------------------------------------------------------------------------------------------------

6.1.12.17 LslRefreshListStart

Diese Funktion startet den eigentlichen Refresh aller Einträge. Ein einmaliger Aufruf dieser Funktion ist nach erfolgreichem Eintragen mit der Funktion [LslRefreshListAdd\(\)](#) notwendig. Ein einmaliger Aufruf nach mehreren Aufrufen von [LslRefreshListAdd\(\)](#) startet ebenfalls die gesamte Refresh-Liste.



Es ist darauf zu achten, welche der beiden Refresh-Listen angesprochen wird.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListStart(int iRLB, DWORD dwFlag);
```

Übergabeparameter	Typ	Beschreibung	
iRLB	INT	ID der Refresh-Liste	
dwFlag	DWORD	LSL_RL_FLAG_DYNAMIC_LIST oder LSL_RL_FLAG_PERMANENT_LIST	
		Beschreibung	
		TRUE	Funktion erfolgreich
		FALSE	Fehler
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.	

6.1.12.18 LslSetDbgLevelEx

Ändert den Debug-Level der lasal32.dll oder der Refresh-Liste auf den angegebenen Wert, wenn der neue Level ≥ 0 ist. Rückgabewert ist der alte Debug-Level.

```
extern "C" int32 LASAL32_EXPORTS LslSetDbgLevelEx(
    int32_t    level,
    uint32_t   dwFlags
);
```

Übergabeparameter	Typ	Beschreibung	
level	int32_t	Neuer Debug-Level ≥ 0	

dwFlags	uint32_t	DBGLEVELLOG_LASAL32: Debug-Level der lasal32.dll ändern DBGLEVELLOG_REFRESHLIST: Debug-Level der Refresh-Liste ändern Achtung: es darf immer nur ein Flag gesetzt sein. Wenn beide Flags gesetzt sind, dann wird nur der Debug-Level der lasal32.dll geändert.
Rückgabeparameter	Typ	Beschreibung
		Alter Debug-Level

6.1.13 Interne Funktionen

6.1.13.1 LslDownloadOS

Entspricht der Funktion [LslDownloadOS\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromClt(
    uint32_t objAddr,
    uint32_t *dValue
);
```

6.1.13.2 LslDownloadOSH

Diese Funktion installiert das angegebene Betriebssystem auf der SPS.

```
typedef uint32_t __cdecl CB_FUNCTYPE2(void *, uint32_t val, uint32_t state);

extern "C" LSL_BOOL LASAL32_EXPORTS LslDownloadOSH(
    int32_t          ocbNum,
    const char       *fileSrc,
    CB_FUNCTYPE2    *callback,
    void            *pUser
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
fileSrc	CONST CHAR	Dateiname der neuen OS Download Datei
Callback	CB_FUNCTYPE2*	Zeiger auf eine CB_FUNCTYPE2 Callback-Funktion. Bei der Übertragung der Datei ist es möglich, dass die Daten in Blöcke unterteilt werden. Die Callback-Funktion wird vor der Übertragung der Datei einmal mit der Dateilänge in Byte und einmal mit der Anzahl der verbleibenden Blöcke als Parameter aufgerufen. Die Callback-Funktion wird während der Übertragung der Datei mit der Anzahl der Byte, die schon kopiert wurden als Parameter aufgerufen.

		Die Callback-Funktion kann verwendet werden, um z.B. einen Fortschrittsbalken zu aktualisieren. Wenn der Parameter callback NULL ist, wird keine Callback-Funktion aufgerufen.
pUser	VOID*	Anwenderparameter, der an die Callback-Funktion übergeben wird

6.1.14 LsIDirect

Mit den LsIDirect-Kommandos ist es möglich, eine einfache Verbindung über TCP/IP zu der Klasse LsIDirect an der SPS aufzubauen. Es sind bis zu 100 Online-Verbindungen (Direct Online Blöcke (DOBs)) möglich.

6.1.14.1 LsIDirectCreateDOB

Legt ein DOB an und liefert den Index auf diesen Block zurück, wenn erfolgreich.

```
extern "C" int32_t LsIDirectCreateDOB();
```

Übergabeparameter	Typ	Beschreibung
keine		
Rückgabeparameter	Typ	Beschreibung
		<p>≥0 Gültiger DOB</p> <p>-1 Ein Fehler ist aufgetreten (verwende GetLastError() Funktionalität)</p>

6.1.14.2 LsIDirectDestroyDOB

Gibt die zuvor mit [LsIDirectCreateDOB\(\)](#) angelegte Ressource wieder frei. Wenn die Verbindung noch nicht geschlossen ist, wird diese geschlossen.

```
extern "C" int32_t LsIDirectDestroyDOB(int32_t nDOB);
```

Übergabeparameter	Typ	Beschreibung
nDOB	int32_t	nDOB von LsIDirectCreateDOB()
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich

		#0 Fehler-Code
--	--	----------------

6.1.14.3 [LslDirectIsOnline](#)

Prüft, ob die mit nDOB übergebene Ressource noch verbunden ist.

```
extern "C" LSL_BOOL LslDirectIsOnline(int32_t nDOB);
```

Übergabeparameter	Typ	Beschreibung	
nDOB	int32_t	Gültiger DOB von LslDirectCreateDOB()	
		FALSE	Nicht online oder ein Fehler ist aufgetreten
		TRUE	Online OK

6.1.14.4 [LslDirectOffline](#)

Beendet die Verbindung, welche mit [LslDirectOnline\(\)](#) aufgebaut wurde. (Die Resource wird hier nicht freigegeben; um die belegte Ressource wieder freizugeben, muss noch [LslDirectDestroyDOB\(\)](#) aufgerufen werden.)

```
extern "C" int32_t LslDirectOffline(int32_t nDOB);
```

Übergabeparameter	Typ	Beschreibung	
nDOB	int32_t	Gültiger DOB von LslDirectCreateDOB()	
		0	Funktion erfolgreich
		#0	Fehler-Code

6.1.14.5 [LslDirectOnline](#)

Baut eine Verbindung mit einem TCP/IP auf. Falls pCallback nicht NULL ist, wird diese Funktion aufgerufen, wenn Daten empfangen wurden, ansonsten siehe [LslDirectReceive\(\)](#).

```
extern "C" int32_t LslDirectOnline(
    int32_t
        nDOB,
```

```
const char          *lpTcp,
uint32_t           nPort,
DO_CALLBACK_FUNCTYPE *pCallback,
Void               *pCookie
);
```

Übergabeparameter		Typ	Beschreibung				
nDOB		int32_t	nDOB von LslDirectCreateDOB()				
lpTcp		const char	Format des Strings TCP:10.10.115.43				
nPort		uint32_t	Ist nPort 0, wird der Standard-Port verwendet				
pCallback		DO_CALLBACK_FUNCTYPE	Zeiger auf eine Callback-Funktion				
pCookie		Void	Anwenderdaten, die mit dem Callback übertragen werden				
Rückgabeparameter		Typ	Beschreibung				
			<table border="1" style="width: 100%; text-align: center;"> <tr> <td>≥0</td><td>Gueltiger DOB</td></tr> <tr> <td>-1</td><td>Fehler</td></tr> </table>	≥0	Gueltiger DOB	-1	Fehler
≥0	Gueltiger DOB						
-1	Fehler						

6.1.14.6 LslDirectReceive

Liefert die empfangenen Daten vom Buffer zurück.

```
extern "C" int32_t LslDirectReceive(int32_t nDOB, uint32_t *pnCmd, uint32_t *pnRealSize, uint8_t dwFlags);

#define LSL_DO_RECEIVE_NOWAIT 0x00000001
// gibt ERROR_SIGMA32_DO_RECEIVE_NO_DATA zurück, wenn keine Daten verfügbar sind

#define LSL_DO_RECEIVE_NOTIMEOUT0x00000002
// Warten auf Daten mit keinem timeout (default 10 sek.)
```

Übergabeparameter		Typ	Beschreibung
nDOB		int32_t	Gueltiger DOB von LslDirectCreateDOB()
pnCmd		uint32_t	Befehl
pnRealSize		uint32_t	Länge der empfangenen Daten
pBuffer		uint8_t	Puffer der die Daten aufnehmen wird
nBufSize		uint32_t	Größe des Datenpuffers
dwFlags		uint32_t	Siehe #define
Rückgabeparameter		Typ	Beschreibung

		0 Funktion erfolgreich ≠0 Fehler-Code
--	--	------------------------------------------

6.1.14.7 [LslDirectReceiveCount](#)

Liefert die Anzahl der Datenpakete zurück, welche sich im Puffer befinden.

```
extern "C" int32_t LslDirectReceiveCount(int32_t nDOB, uint32_t *pCount);
```

Übergabeparameter	Typ	Beschreibung
nDOB	int32_t	Gültiger DOB von LslDirectCreateDOB()
pCount	uint32_t	Datenpaket-Zähler
Übergabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich ≠0 Fehler-Code

6.1.14.8 [LslDirectSend](#)

Übermittelt ein Kommando mit den nötigen Daten im pSend an das Ziel.

```
extern "C" int32_t LslDirectSend(int32_t nDOB, uint32_t nCmd, const uint8_t *pSend, uint32_t
```

Übergabeparameter	Typ	Beschreibung
nDOB	int32_t	Gültiger DOB von LslDirectCreateDOB()
nCmd	uint32_t	Send Kommando-ID
pSend	const uint8_t	Zeiger auf die zu sendenden Daten
nSendLen	uint32_t	Größe der Sendedaten
Übergabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich ≠0 Fehler-Code

6.1.14.9 LslDirectSetParam

Damit kann der Timeout von [LslDirectReceive\(\)](#) gesetzt werden.

```
extern "C" int32_t LslDirectSetParam(int32_t nDOB, uint32_t nType, uint32_t nData);
#define LSL_DO_SET_TYPE_TIMEOUT_RECEIVE 0
// Daten us Timeout-Wert in millisekunden
```

Übergabeparameter		Typ	Beschreibung	
nDOB		int32_t	Gültiger DOB von LslDirectCreateDOB()	
nType		uint32_t	Parametertyp (siehe LSL_DO_SET_TYPE_)	
nData		uint32_t	Parameterdaten	
Rückgabeparameter		Typ	Beschreibung	
			0	Funktion erfolgreich
			#0	Fehler-Code

6.1.15 Anhang A

CPU Status-Codes

0	RUN RAM
1	RUN ROM
2	RUNTIME ERROR
3	POINTER ERROR
4	CHECKSUM ERROR
5	WATCHDOG TIMEOUT
6	GENERAL ERROR
7	PROM DEFECT
8	RESET
9	WATCHDOG DEFECT
10	STOP
11	PROG BUSY

- | | |
|----|---------------------|
| 12 | PROGRAM LENGTH |
| 13 | PROGRAM END |
| 14 | PROGRAM MEMO |
| 15 | STOP BREAKPOINT |
| 16 | CPU STOP |
| 17 | INTERRUPT ERROR |
| 18 | SINGLE STEP |
| 19 | READY |
| 20 | LOAD |
| 21 | WRONG MODULE |
| 22 | MEMORY FULL |
| 23 | NOT LINKED |
| 24 | DIVIDE BY ZERO |
| 25 | DIAS ERROR |
| 26 | WAIT |
| 27 | OPSYS PROGRAM |
| 28 | OPSYS INSTALLED |
| 29 | OPSYS TOO LONG |
| 30 | NO OPERATING SYSTEM |
| 31 | SEARCH OPSYS |
| 32 | NO DEVICE |
| 33 | UNUSED CODE |
| 34 | MEMORY ERROR |
| 35 | MAX IO |
| 36 | MODUL LOAD ERROR |
| 37 | BOOTIMAGE FAILURE |
| 38 | ERROR 38 |
| 39 | ERROR 39 |
| 40 | APPL LOAD FROM DISK |

41 APPL SAVE TO DISK

42 ERROR 42

43 ERROR 43

44 ERROR 44

45 ERROR 45

46 ERROR LOADING LOADER

47 ERROR SAVING PROJECT

48 ERROR 48

49 ERROR 49

50 ACCESS EXCEPTION

51 BOUND EXCEEDED

52 PRIVLEDGED INSTR

53 ERROR 53

54 ERROR 54

55 ERROR 55

56 ERROR 56

57 ERROR 57

58 ERROR 58

59 ERROR 59

60 ERROR 60

61 ERROR 61

62 ERROR 62

63 ERROR 63

64 INTERNAL ERROR

65 FILE ERROR

66 ERROR 66

67 ERROR 67

68 ERROR 68

69 ERROR 69

70	ERROR 70
71	ERROR 71
72	ERROR 72
73	ERROR 73
74	ERROR 74
75	ERROR 75
76	ERROR 76
77	ERROR 77
78	ERROR 78
79	ERROR 79
80	ERROR 80
81	ERROR 81
82	ERROR 82
83	ERROR 83
84	ERROR 84
85	ERROR 85
86	ERROR 86
87	ERROR 87
88	ERROR 88
89	ERROR 89
90	ERROR 90
91	ERROR 91
92	ERROR 92
93	ERROR 93
94	ERROR 94
95	ERROR 95
96	ERROR 96
97	ERROR 97
98	RETURN FROM SCR

99	ERROR 99
100	ERROR 100
101	ERROR 101
102	ERROR 102
103	ERROR 103
104	ERROR 104
105	ERROR 105
106	ERROR 106
107	ERROR 107
108	ERROR 108
109	ERROR 109
110	ERROR 110
111	ERROR 111
112	ERROR 112
113	ERROR 113
114	ERROR 114
115	ERROR 115
116	ERROR 116
117	ERROR 117
118	ERROR 118
119	ERROR 119
120	ERROR 120
121	ERROR 121
122	ERROR 122
123	ERROR 123
124	ERROR 124
125	ERROR 125
126	ERROR 126
127	ERROR 127

128	ERROR 128
129	ERROR 129
130	ERROR 130
131	ERROR 131
132	ERROR 132
133	ERROR 133
134	ERROR 134
135	ERROR 135
136	ERROR 136
137	ERROR 137
138	ERROR 138
139	ERROR 139
140	ERROR 140
141	ERROR 141
142	ERROR 142
143	ERROR 143
144	ERROR 144
145	ERROR 145
146	ERROR 146
147	ERROR 147
148	ERROR 148
149	ERROR 149
150	ERROR 150
151	ERROR 151
152	ERROR 152
153	ERROR 153
154	ERROR 154
155	ERROR 155
156	ERROR 156

157	ERROR 157
158	ERROR 158
159	ERROR 159
160	ERROR 160
161	ERROR 161
162	ERROR 162
163	ERROR 163
164	ERROR 164
165	ERROR 165
166	ERROR 166
167	ERROR 167
168	ERROR 168
169	ERROR 169
170	ERROR 170
171	ERROR 171
172	ERROR 172
173	ERROR 173
174	ERROR 174
175	ERROR 175
176	ERROR 176
177	ERROR 177
178	ERROR 178
179	ERROR 179
180	ERROR 180
181	ERROR 181
182	ERROR 182
183	ERROR 183
184	ERROR 184
185	ERROR 185

186	ERROR 186
187	ERROR 187
188	ERROR 188
189	ERROR 189
190	ERROR 190
191	ERROR 191
192	ERROR 192
193	ERROR 193
194	ERROR 194
195	ERROR 195
196	ERROR 196
197	ERROR 197
198	ERROR 198
199	ERROR 199
200	ERROR 200
201	ERROR 201
202	ERROR 202
203	ERROR 203
204	ERROR 204
205	ERROR 205
206	ERROR 206
207	ERROR 207
208	ERROR 208
209	ERROR 209
210	ERROR 210
211	ERROR 211
212	ERROR 212
213	ERROR 213
214	ERROR 214

215	ERROR 215
216	ERROR 216
217	ERROR 217
218	ERROR 218
219	ERROR 219
220	ERROR 220
221	ERROR 221
222	ERROR 222
223	ERROR 223
224	LINKING
225	LINKER ERROR"
226	LINKING DONE
227	ERROR 227
228	ERROR 228
229	ERROR 229
230	ERROR 230
231	OP BURN FAIL
232	INSTALLING OS...
233	ERROR 233
234	ERROR 234
235	ERROR 235
236	ERROR 236
237	ERROR 237
238	ERROR 238
239	ERROR 239
240	WAIT FOR POWERDOWN
241	REBOOTING...
242	RAM SAVE
243	RAM LOAD

244	ERROR 244
245	ERROR 245
246	ERROR 246
247	ERROR 247
248	ERROR 248
249	ERROR 249
250	ERROR 250
251	ERROR 251
252	ERROR 252
253	PRE-RUN STATE
254	PRE-RESET STATE

6.1.15.1 LslExecNewInstr

Entspricht der Funktion [LslExecNewInstrH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslExecNewInstr(
    uint32_t dwSrvAddress,
    uint16_t wCmd,
    uint32_t adwParas[],
    uint32_t dwParaCnt,
    uint8_t *pResult,
    uint32_t dwResultCnt,
    uint32_t useLenField
);
```

Beispiel

Das LASAL-Projekt liefert ein NewInstr-Befehlsaufruf, mit dem Daten aus einem WORD (UINT) Array gelesen werden können.

Der Befehl ist wie folgt konfiguriert:

Command Byte	1
Parameter 0	StartIndex in Array
Parameter 1	Anzahl der zu lesenden Einträge

LASAL CLASS

```
FUNCTION VIRTUAL GLOBAL Subscription::NewInst
```

```

VAR_INPUT
    pPara      : ^CmdStruct;
    pResult    : ^Results;
END_VAR
VAR_OUTPUT
    ret_code    : IprStates;
END_VAR
VAR
    ReadIndex : DINT;
    Anzahl    : DINT;
END_VAR

// this function belongs to the ONLINE interface and can also be used in the
Program

    ret_code:= READY;      // return code is o.k.

CASE pPara^.uIcmcmd OF

    0:
        // function 0 brings trend status
        // -> <BufferSize><Füllstand>
        pResult^.UiLng:= 3*SizeOF(UDINT)+2;// Lng = payloadLng +2
        (pResult+2)^$DINT      := sizeOfBuffer;
        (pResult+2+4)^$DINT    := offset;

    1:
        // function 1 data from number (max. 250 bytes per inquiry) thus 125 entries
        // <bring data from trend> <StartIndex> <number>

        ReadIndex  := pPara^.aPara[0];
        number     := pPara^.aPara[1];

        // -- Check on invalid calls
        IF (ReadIndex < (sizeOfBuffer/SIZEOF(INT))) & (number <= 250/Sizeof(INT)) THEN
            pResult^.UiLng :=2; // length same
            pResult^.UiLng += number$UINT*Sizeof (INT); // not like 125 entries anymore
            _memcpy(#pResult^.aData, pData+ (ReadIndex * sizeof (INT)),
Anzahl$UDINT* sizeof (INT));
        ELSE
            ret_code:= ERROR;
        END_IF;

        // all other commands are still free e.g. delete buffer or start aso.
        2:
            ret_code:= ERROR; // must be READY if used!!

            ELSE
            // unknown command
            ret_code:= ERROR;
        END_CASE

END_FUNCTION //VIRTUAL GLOBAL Subscription::NewInst

```

C Program

```
-----  
// GetTrendData  
//  
// Reads data from the trend buffer  
//  
Parameter:  
// readIndex .. Index (on an entry) in the trend buffer  
Number of entries to read  
// resultBuf .. Result buffer  
// sizeOfBuf .. Size of the result buffer  
//  
// Return value:  
// >= 0 .. Number of entries read  
// < 0 .. Error  
-----  
int GetTrendData(DWORD *addr,  
DWORD readIndex,  
DWORD anzahl,  
WORD *resultBuf,  
DWORD sizeOfBuf)  
{  
ChMode chmode;  
char clsName[300];  
DWORD paras[2];  
DWORD requestedBytes;  
DWORD returnedBytes;  
  
BYTE cmdBuf[256];  
DWORD addr;  
  
if (!LslGetObject("Aufzeichnung0", &addr, &chmode, clsName))  
{  
printf("Error: LslGetObject failed !\n");  
return -1;  
}  
  
requestedBytes = 2 + anzahl * sizeof(WORD);  
if (requestedBytes > sizeof(cmdBuf))  
{  
printf("Error: GetTrendData - ungultige Parameter !\n");  
return -1;  
}  
  
paras[0] = readIndex;  
paras[1] = anzahl;  
  
if (!LslExecNewInstr(addr,  
1, // wCmd,  
paras, // adwParas  
sizeof(paras), // dwParaCnt  
cmdBuf,  
requestedBytes, // dwResultCnt
```

```

0 // useLenField
        ))
{
printf("Error: LslExecNewInstr failed !\n");
return -1;
}
returnedBytes = (int)*(WORD *)cmdBuf;
if (returnedBytes != requestedBytes)
printf("GetTrendData: returnedBytes != requestedBytes\n");

if (returnedBytes >= 2)
{
returnedBytes -= 2; // discount length field
returnedBytes = __min(returnedBytes, sizeOfBuf);
memcpy(resultBuf, &cmdBuf[2], returnedBytes);
}
else
{
printf("GetTrendData: ungultiges Laengenfeld erhalten!\n");
returnedBytes = 0;
return -1;
}

return (returnedBytes / sizeof(WORD));
}

```

6.1.15.2 LslExecNewInstrH

Ruft die Methode NewInstr (= command call) eines Objektes auf.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslExecNewInstrH(
    int32_t    ocbNum,
    uint32_t   dwSrvAddress,
    uint16_t   wCmd,
    uint32_t   adwParas[],
    uint32_t   dwParaCnt,
    uint8_t    *pResult,
    uint32_t   dwResultCnt,
    uint32_t   useLenField
);

```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
dwSrvAddress	uint32_t	Server-Adresse
wCmd	uint16_t	Kommando
adwParas	uint32_t	Befehl-Parameter (Array mit 4-Byte Werte).
dwParaCnt	uint32_t	Anzahl der Befehlsparameter. Maximal 20 Parameter sind möglich.

pResult	uint8_t*	Result buffer Die ersten 2 Byte enthalten die Gesamtlänge des Ergebnispuffers (inkl. Längenfeld). Eine maximale Länge von 252 Byte ist möglich.
dwResultCnt	uint32_t	Größe des Ergebnispuffers in Byte
useLenField	uint32_t	Dieser Parameter wird $\neq 0$, wenn das Längenfeld in der Rückmeldung interpretiert werden soll und nur so viele Byte in den Ergebnispuffer kopiert werden sollen, wie im Längenfeld definiert sind. Ist dieser Parameter 0, wird dwResultCnt für die Anzahl der zu kopierenden Byte verwendet.
Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich FALSE Fehler

Mit Befehlsaufrufen können Objektbefehle ausgelöst werden. Die Befehle werden in der NewInstr-Methode der Klasse durchgeführt. Befehlsaufrufe werden oft zur Datenübertragung verwendet.

6.1.15.3 [LslGetObjCls](#)

Entspricht der Funktion [LslGetObjClsH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjCls(
    const uint32_t objAddr,
    char *className
);
```

6.1.15.4 [LslGetObjClsH](#)

Ermittelt den Klassennamen eines Objekts.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjClsH(
    int32_t ocbNum,
    const uint32_t objAddr,
    char *className
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
objAddr	const uint32_t	Objekt-Adresse
className	char	Name der Klasse

Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich
		FALSE Fehler

Beispiel

```
static char* GetChannelTypeStr(ChMode Mode)
{
    static char object[] = "object";
    static char command_server[] = "command server";
    static char data_server[] = "data server";
    static char command_client[] = "command client";
    static char data_client[] = "data client";
    static char object_client[] = "object client";
    static char unknown[] = "unknown type";

    switch (Mode)
    {
        case _CH_OBJ:
            return (object);
        break;

        case _CH_CMD:
            return (command_server);
        break;

        case _CH_SVR:
            return (data_server);
        break;

        case _CH_CLT_CMD:
            return (command_client);
        break;

        case _CH_CLT_DATA:
            return (data_client);
        break;

        case _CH_CLT_OBJ:
            return (object_client);
        break;

        default:
            return (unknown);
        break;
    }

    static void TestObjectFunc()
    {
```

```

uint32_t      ObjAddr;
char          ClassName1[512];
char          ClassName2[512];
ChModeMode;

if (::Online("10.10.170.190", 0, 0, 0, 0))
{
    if (LslGetObject("ClassBigData1", &ObjAddr, &Mode, ClassName1))
    {
        if (LslGetObjCls(ObjAddr, ClassName2))
        {
            printf("Class names of %s is %s %s\n", " ClassBigData1",ClassName1,
                   ClassName2);
            printf("Object type is <%s>\n", GetChannelTypeStr(Mode));
        }
        else
        {
            printf("LslGetObjCls() failed 0x%08X\n", GetLastError());
        }
    }
    else
    {
        printf("LslGetObject() failed 0x%08X\n", GetLastError());
    }
}
else
{
    printf("Error Online:0x%08X \n", GetLastError());
}

::Offline();
}

```

6.1.15.5 LslGetObject

Entspricht der Funktion [LslGetObjectH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObject(
    const char  *objName,
    uint32_t     *objAddr,
    ChMode       *pMode,
    char         *pszClsName = NULL
);

```

6.1.15.6 LslGetObjectEx

Entspricht LslGetObjectExH() mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjectEx(
    const char  *objName,
    plcptr_t    *objAddr,
    ChMode       *pMode,
    char         *pszClsName,
    uint8_t      *pDataBufferFlag
);

```

6.1.15.7 LslGetObjectExH

Diese Funktion ermittelt eine Objekt- oder Kanaladresse (Kanal ist ein Client oder Server) und bei Objekten des Typs _CH_OBJ auch den Namen der Klasse.

Mit der Funktion LslGetAdressVar() können die Adressen von globalen Variablen ermittelt werden.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjectExH(
    int32_t          ocbNum,
    const char      *objName,
    plcptr_t        *objAddr,
    ChMode          *pMode,
    char            *pszClsName,
    uint8_t          *pDataBufferFlag
);
```

Übergabeparameter		Typ	Beschreibung
ocbNum		int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
objName		const char*	Name des Objekts oder Kanals (ObjectName.ServerName oder ObjectName.ClientName) als 0-terminierter ASCII-String
objAddr		plcptr_t*	Zeiger auf eine Variable, wo die Objekt- oder Kanaladresse eingetragen wird
pMode		ChMode*	Zeiger auf eine Variable, wo der Kanaltyp eingetragen wird. Werte siehe ChMode in Lasal32.h
pszClsName		char*	Zeiger auf einen Puffer, wo der Name der Klasse eingetragen wird, wenn es sich um ein Objekt des Typs (_CH_OBJ) handelt (0-terminierter ASCII-String). Dieser Parameter darf NULL sein, wenn der Klassennname nicht benötigt wird.
pDataBufferFlag		uint8_t*	Zeiger auf eine Variable, wo eingetragen wird, ob dieser Kanal einen Datenpuffer besitzt (DataBufferFlag=4) oder nicht (DataBufferFlag=0). Bei einem Kanal mit Datenpuffer (String-Server) werden mehr als 4 Bytes an Daten übertragen, wenn dieser Kanal zur Refresh-Liste hinzugefügt wird. Dieser Parameter darf NULL sein, wenn die Information nicht benötigt wird.
Rückgabeparameter		Typ	Beschreibung
			<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>

Beispiel

```

BOOL bResult = FALSE;
ChMode mode;
uint32_t dwAddr = 0;
char szClsName[256] = { 0 };
uint8_t DataBufferFlag;

bResult = LslGetObjectEx("ObjectName.ServerName", &dwAddr, &mode, szClsName, &DataBufferFlag
if(bResult == TRUE && dwAddr
{
    //Gültige Adresse in dwAddr
}
else
    printf("LslGetObjectEx failed(0x%08X)\n", GetLastError());

```

6.1.15.8 LslGetObjectH

Diese Funktion ermittelt eine Objekt- oder Kanaladresse (Kanal ist ein Client oder Server) und bei Objekten des Typs `_CH_OBJ` auch den Namen der Klasse.

Mit der Funktion `LslGetAdressVar()` können die Adressen von globalen Variablen ermittelt werden.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjectH(
    int32_t      ocbNum,
    const char   *objName,
    uint32_t      *objAddr,
    ChMode       *pMode,
    char         *pszClsName = NULL
);

```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
objName	const char*	Name des Objekts oder Kanals (ObjectName.ServerName oder ObjectName.ClientName) als 0-terminierter ASCII-String
objAddr	uint32_t*	Objekt- oder Kanaladresse
pMode	ChMode*	Kanaltyp; Werte siehe ChMode in Lasal32.h
pszClsName	CHAR*	Name der Klasse, wenn es sich um ein Objekt des Typs <code>_CH_OBJ</code> handelt
Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich

		FALSE Fehler
--	--	--------------

Beispiel

Werte siehe ChMode in Lasal32.h.

Siehe LslGetObjCls().

6.1.15.9 LslGetObjectID

Entspricht LslGetObjectIDH() mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjectID(
    const char *name,
    uint32_t *adresse
);
```

6.1.15.10 LslGetObjectIDH

Diese Funktion ermittelt die Adresse des angegebenen LASAL-Objekts.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjectIDH(
    int32_t ocbNum,
    const char *name,
    uint32_t *adresse
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
Name	const char*	Zeiger auf den Namen des Objekts (ASCII-0-String)
adresse	uint32_t*	Zeiger auf eine Variable, wo die Objekt-Adresse eingetragen wird
Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich
		FALSE Fehler
		Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.

Beispiel

```
CStringA ObjName;
uint32_t ObjAddr;
```

```

printf("Object Name\n");
if (!InputString(ObjName))
    return;

if (LslGetObjectID(ObjName, &ObjAddr))
    printf("Object Address: 0x%08X \n", ObjAddr);
else
    printf("LslGetObjectID() failed(0x%08X)\n", GetLastError());

```

6.1.15.11 LslReadFromClt

Entspricht der Funktion [LslReadFromCltH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromClt(
    uint32_t objAddr,
    uint32_t *dvalue
);

```

6.1.15.12 LslReadFromCltH

Diese Methode ruft die Read-Methode eines Servers auf, der mit einem Client verbunden ist. Die Adresse des Clients muss angegeben werden und der gelesene Wert wird auf den Server geschrieben.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromCltH(
    int32_t ocbNum,
    uint32_t objAddr,
    uint32_t *dvalue
);

```

Übergabeparameter	Typ	Beschreibung	
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
objAddr	uint32_t	Kanaladresse des Clients	
dValue	uint32_t*	Zeiger auf eine Variable, wo der Rückgabewert der Read-Methode eingetragen wird	
Rückgabeparameter	Typ	Beschreibung	
	LSL_BOOL	TRUE	Funktion erfolgreich
		FALSE	Fehler

Beispiel

```
static char* GetChannelTypeStr(ChMode Mode); //Siehe LslGetObjCls()
```

```

static void TestReadFromClt()
{
    uint32_t      ObjAddr;
    ChModeMode;
    uint32_t      Value;

    if (::Online("10.10.170.190", 0, 0, 0, 0))
    {
        if (LslGetObject("DataClass1.DataClient", &ObjAddr, &Mode, NULL))
        {
            if (Mode == _CH_CLT_CMD || Mode == _CH_CLT_DATA || Mode == _CH_CLT_OBJ)
            {
                if (LslReadFromClt(ObjAddr, &Value))
                {
                    printf("Client type is <%s>\n", GetChannelTypeStr(Mode));
                    printf("Client value is %u\n", Value);
                }
                else
                {
                    printf("LslReadFromClt() failed 0x%08X\n", GetLastError());
                }
            }
            else
            {
                printf("No Client given\n");
            }
        }
        else
        {
            printf("LslGetObject() failed 0x%08X\n", GetLastError());
        }
    }
    else
    {
        printf("Error Online:0x%08X \n", GetLastError());
    }
    ::Offline();
}

```

6.1.15.13 LslReadFromSvr

Entspricht der Funktion [LslReadFromSvrH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromSvr(
    uint32_t  objAddr,
    uint32_t  *dValue
);

```

6.1.15.14 LslReadFromSvrH

Ruft die Read-Methode eines Servers auf. Diese Funktion kann nur für numerische Server verwendet werden. Für String-Server verwenden Sie bitte die Funktion [LslReadFromSvrStrH\(\)](#).

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromSvrH(
```

```
    int32_t      ocbNum,
    uint32_t     objAddr,
    uint32_t     *dValue
);
```

Übergabeparameter		Typ	Beschreibung	
ocbNum		int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())	
objAddr		uint32_t	Serverkanal-Adresse	
dValue		uint32_t	Rückgabewert der Read()-Methode	
Rückgabeparameter		Typ	Beschreibung	
			TRUE	Funktion erfolgreich
			FALSE	Fehler

Beispiel

```
static char* GetChannelTypeStr(ChMode Mode); //Siehe LslGetObjCls()

static void TestReadFromSvr()
{
    uint32_t          ObjAddr;
    ChModeMode;
    uint32_t          Value;

    if (::Online("10.10.170.190", 0, 0, 0, 0))
    {
        if (LslGetObject("DataClass1.ClassSvr", &ObjAddr, &Mode, NULL))
        {
            if (Mode == _CH_CMD || Mode == _CH_SVR)
            {
                if (LslReadFromSvr(ObjAddr, &Value))
                {
                    printf("Server type is <%s>\n", GetChannelTypeStr(Mode));
                    printf("Server value is %u\n", Value);
                }
                else
                {
                    printf("LslReadFromSvr() failed 0x%08X\n", GetLastError());
                }
            }
            else
                printf("No Server given\n");
        }
        else
            printf("LslGetObject() failed 0x%08X\n", GetLastError());
    }
}
```

```

    }
else
    printf("Error Online:0x%08X \n", GetLastError());
    ::Offline();
}
}

```

6.1.15.15 LslReadFromSvrStr

Entspricht LslReadFromSvrStrH() mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromSvrStr(
    plcptr_t    objAddr,
    char        *pString,
    uint32_t    dwLen,
    uint32_t    *pStringLen
);

```

6.1.15.16 LslReadFromSvrStrH

Diese Funktion ruft die NewInst-Methode eines String-Servers auf und gibt den String und die String-Länge zurück.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromSvrStrH(
    int32_t    ocbNum,
    plcptr_t    objAddr,
    char        *pString,
    uint32_t    dwLen,
    uint32_t    *pStringLen
);

```

Übergabeparameter		Typ	Beschreibung
ocbNum		int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
objAddr		plcptr_t	Serverkanal-Adresse
pString		char*	Zeiger auf einen Puffer, wo der 0-terminierte ASCII- oder Unicode-String eingetragen wird
dwLen		uint32_t	Puffergröße in Byte. Wegen der Null-Terminierung muss der Puffer um ein Byte (ASCII) oder zwei Byte (Unicode) größer sein als die String-Länge.
pStringLen		uint32_t*	Zeiger auf eine Variable, wo die String-Länge in Byte eingetragen wird (ohne Null-Terminierung). Dieser Zeiger darf NULL sein, wenn die Länge nicht benötigt wird.
Rückgabeparameter		Typ	Beschreibung
			TRUE Funktion erfolgreich

		FALSE Fehler
Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.		

Beispiel

```
uint32_t dwAddr = 0;
ChMode mode;
char strServer1[128];

strcpy(strServer1, "ObjectName.ServerName");

if (LslGetObject(strServer1, &dwAddr, &mode, NULL) == FALSE)
    printf("LslGetObject failed(0x%08X)\n", GetLastError());
else if (dwAddr == 0)
    printf("Object/Server(%s) not found\n", strServer1);
else if (mode != _CH_CMD)
    printf("Wrong Type(%s)\n", strServer1);
else
{
    char buffer[1024];
    unsigned int len = 0;
    if (LslReadFromSvrStr(dwAddr, buffer, 1000, &len) == FALSE)
        printf("LslReadFromSvrStr failed(0x%08X)\n", GetLastError());
    else
    {
        buffer[len] = 0;
        printf("Len:%d\n", len);
        printf("Read:'%s'", buffer);
    }
}
```

6.1.15.17 LslWriteToClt

Entspricht der Funktion [LslWriteToCltH\(\)](#) mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToClt(
    uint32_t    objAddr,
    uint32_t    dvalue
);
```

6.1.15.18 LslWriteToCltH

Diese Funktion ruft die Write()-Methode eines Servers auf, der mit einem Client verbunden ist. Die Adresse des Clients ist anzugeben und der Wert des Servers wird geändert.

```
extern "C" bool LslWriteToClt(
    DWORD    objAddr,
```

```
    DWORD  dValue
} ;
```

Übergabeparameter		Typ	Beschreibung	
objAddr		DWORD	Kanaladresse des Clients	
dValue		DWORD	Wert, der auf den Client geschrieben werden soll	
Rückgabeparameter		Typ	Beschreibung	
			TRUE Funktion erfolgreich	
			FALSE Fehler	

Beispiel

```
static char* GetChannelTypeStr(ChMode Mode); //Siehe LslGetObjCls()

static void TestWriteToClt()
{
    uint32_t      ObjAddr;
    ChModeMode;
    char         strValue[256];
    uint32_t      Value;

    printf("Client value is = ");
    scanf("%255s", strValue);
    Value = atoi(strValue);

    if (::Online("10.10.170.190", 0, 0, 0, 0))
    {
        if (LslGetObject("DataClass1.CmdClient", &ObjAddr, &Mode, NULL))
        {
            printf("Channel type is <%s>\n", GetChannelTypeStr(Mode));
            if (Mode == _CH_CLT_CMD || Mode == _CH_CLT_DATA || Mode == _CH_CLT_OBJ)
            {
                if (LslWriteToClt(ObjAddr, Value))
                {
                    Value = 0;
                    if (LslReadFromClt(ObjAddr, &Value))
                    {
                        printf("Value read from Client is %u\n", Value);
                    }
                    else
                    {
                        printf("LslReadFromClt() failed 0x%08X\n", GetLastError());
                    }
                }
                else
                {
                    printf("LslWriteToClt() failed 0x%08X\n", GetLastError());
                }
            }
        }
    }
}
```

```

        }
    }
    else
        printf("No Client given\n");
}
else
{
    printf("LslGetObject() failed 0x%08X\n", GetLastError());
}
else
{
    printf("Error Online:0x%08X \n", GetLastError());
    ::Offline();
}

```

6.1.15.19 LslWriteToSvr

Entspricht der Funktion [LslWriteToSvrH\(\)](#) mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvr(
    uint32_t    objAddr,
    uint32_t    dValue
);

```

6.1.15.20 LslWriteToSvrEx

Entspricht LslWriteToSvrExH() mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrEx(
    plcptr_t    objAddr,
    uint32_t    dValue,
    uint32_t    *pResult
);

```

6.1.15.21 LslWriteToSvrExH

Diese Funktion ruft über die NewInst-Methode eines Servers seine Write()-Methode auf, um den Serverwert zu ändern.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrExH(
    int32_t      ocbNum,
    plcptr_t    objAddr,
    uint32_t    dValue,
    uint32_t    *pResult
);

```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())

objAddr	plcptr_t	Serverkanal-Adresse
dValue	uint32_t	Neuer Wert, der auf den Server geschrieben werden soll
pResult	uint32_t*	Zeiger auf eine Variable, wo der Rückgabewert der NewInst-Methode eingetragen wird
Rückgabeparameter	Typ	Beschreibung
		<p>TRUE Funktion erfolgreich</p> <p>FALSE Fehler</p> <p>Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion GetLastError() auf.</p>

Beispiel

```
uint32_t dwAddr = 0;
ChMode mode;
uint32_t Result = 0;
char numServer1[128];

strcpy(numServer1, "ObjectName.ServerName");

if (LslGetObject(numServer1, &dwAddr, &mode, NULL) && dwAddr)
{
    if (!LslWriteToSvrEx(dwAddr, 1, &Result))
    {
        printf("Write To Server failed(%d)\n", Result);
    }
}
else
    printf("LslGetObject failed(0x%08X)\n", GetLastError());
```

6.1.15.22 LslWriteToSvrH

Ruft die Write()-Methode eines Servers auf, um den Serverwert zu ändern.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrH(
    int32_t      ocbNum,
    uint32_t     objAddr,
    uint32_t     dValue
);
```

Übergabeparameter	Typ	Beschreibung
ocbNum	int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
objAddr	uint32_t	Serverkanal-Adresse

dValue	uint32_t	Neuer Wert, der auf den Server geschrieben werden soll
Rückgabeparameter	Typ	Beschreibung
		TRUE Funktion erfolgreich FALSE Fehler

Beispiel

```
static char* GetChannelTypeStr(ChMode Mode); //Siehe LslGetObjCls()

static void TestWriteToSvr()
{
    uint32_t          ObjAddr;
    ChMode Mode;
    char              strValue[256];
    uint32_t          Value;

    printf("Server value is = ");
    scanf("%255s", strValue);
    Value = atoi(strValue);

    if (::Online("10.10.170.190", 0, 0, 0, 0))
    {
        if (LslGetObject("DataClass1.ClassSvr", &ObjAddr, &Mode, NULL))
        {
            printf("Channel type is <%s>\n", GetChannelTypeStr(Mode));
            if (Mode == _CH_CMD || Mode == _CH_SVR)
            {
                if (LslWriteToSvr(ObjAddr, Value))
                {
                    Value = 0;
                    if (LslReadFromSvr(ObjAddr, &Value))
                    {
                        printf("Value read from Server is %u\n", Value);
                    }
                    else
                    {
                        printf("LslReadFromSvr() failed 0x%08X\n", GetLastError());
                    }
                }
                else
                {
                    printf("LslWriteToSvr() failed 0x%08X\n", GetLastError());
                }
            }
            else
                printf("No Server given\n");
        }
    }
}
```

```

        printf("LslGetObject() failed 0x%08X\n", GetLastError());
    }
}
else
    printf("Error Online:0x%08X \n", GetLastError());
    ::Offline();
}

```

6.1.15.23 LslWriteToSvrStr

Entspricht LslWriteToSvrStrH() mit ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrStr(
    plcptr_t    objAddr,
    const char  *pString,
    int32_t      iLen
);

```

6.1.15.24 LslWriteToSvrStrH

Diese Funktion ruft die NewInst-Methode eines String-Servers auf, um den String zu schreiben.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrStrH(
    int32_t      ocbNum,
    plcptr_t    objAddr,
    const char  *pString,
    int32_t      iLen
);

```

Übergabeparameter		Typ	Beschreibung
ocbNum		int32_t	Online Connection Block Nummer (Rückgabewert von OcbOpen())
objAddr		plcptr_t	Serverkanal-Adresse
pString		const char*	Zeiger auf den 0-terminierten ASCII- oder Unicode-String, der auf den Server geschrieben werden soll. Wenn hier NULL übergeben wird, dann wird ein Leerstring geschrieben.
iLen		int32_t	Anzahl der Bytes die geschrieben werden sollen oder -1 für den gesamten String
Rückgabeparameter		Typ	Beschreibung
			TRUE Funktion erfolgreich
			FALSE Fehler

Um erweiterte Fehlerinformationen abzurufen, rufen Sie die Funktion [GetLastError\(\)](#) auf.

Beispiel

```
uint32_t dwAddr = 0;
ChMode mode;
char strSend1[128];
char strServer1[128];

strcpy(strSend1, "Display in StringServer1");
strcpy(strServer1, "ObjectName.ServerName");

if (LslGetObject(strServer1, &dwAddr, &mode, NULL) == FALSE)
    printf("LslGetObject failed(0x%08X)\n", GetLastError());
else if (dwAddr == 0)
    printf("Object/Server(%s) not found\n", strServer1);
else if (mode != _CH_CMD)
    printf("Wrong Type(%s)\n", strServer1);
else
{
    if (LslWriteToSvrStr(dwAddr, strSend1, strlen(strSend1)) == FALSE)
        printf("LslWriteToSvrStr failed(0x%08X)\n", GetLastError());
}
```

6.2 Comlink API im Loader

6.2.1 Einleitung

Comlink stellt die Regeln für Visualisierungscomputer (Clients) und SPS (Server) für den Datenaustausch zu Verfügung. Der Client fordert Daten vom Server und dieser schickt die angeforderten Daten als Antwort an den Client.

Der Client sendet entweder eine Anfrage an den Server und wartet auf die Antwort oder er speichert eine Update-Zelle in den Puffer des Update-Servers. Eine Update-Zelle im Update-Puffer weist den Server an, ständig Eingangsdaten in der SPS zu überwachen und Antworten an den Client zu senden, wenn sich der Wert der Eingangsdaten ändert.

Der Server verwaltet zwei verschiedene Update-Puffer: Den statischen und den dynamischen Update-Puffer. Der Client entscheidet, wo die Update-Zellen gespeichert werden. In der Regel wird der statische Update-Puffer mit den Update-Zellen beim Projektstart initialisiert und bleibt dann unverändert. Der Inhalt des dynamischen Update-Puffers wird jedoch geändert, wenn neue Variablen erforderlich sind, z.B. wenn der Benutzer einen Bildschirm mit neuen Variablen auswählt.

6.2.2 Die Comlink-Schnittstelle in LASAL Class

Die Comlink-Schnittstelle in LASAL CLASS besteht aus den folgenden Funktionen:

- Login
- TxCommand
- TxUpd
- StartStopRefresh
- InstallCallback
- Logout
- LDR_SetCanWait
- LDR_SetCanWaitRemote
- LDR_SetRs232ComlinkParams

Zur Kommunikation mit einem Comlink-Server müssen folgende Schritte unternommen werden:

- Einloggen auf dem Server mit der Login-Funktion.
- Anwendungen, die mit dem Command Interpreter des Comlink-Servers kommunizieren wollen, z.B.: zum Datenaustausch eines Objekt-Serverkanals, müssen TxCommand aufrufen.
- Anwendungen, die über Änderungen der Datenwerte eines Objektes benachrichtigt werden wollen, ohne das Objekt abzufragen, sollen die Funktionen TxUpd, StartStopRefresh und InstallCallBack verwenden.

Die Schnittstellen-Funktionen sind in den Loader eingebunden. Wenn die Comlink-Funktionen nicht benötigt werden, kann das Projekt mit einem Loader, der den Comlink-Code oder Teile davon nicht enthält, gespeichert werden. Um einen individuellen Loader zu verwenden, sollten folgende Schritte unternommen werden:

- Öffnen Sie den Projekt-Loader aus dem Unterverzeichnis Loader des LASAL CLASS Programmverzeichnisses.
- Stellen Sie die folgenden Definitionen bei loader.h ein:

```
#define COMLINK_LASAL // auskommentieren, wenn der ganze Comlink-Code
ausgeschlossen werden soll
#define COMLINK_TCP_SERVER // auskommentieren, wenn kein TCP-Comlink-Server
benötigt wird
#define COMLINK_TCP_CLIENT // auskommentieren, wenn kein TCP-Comlink-Client
benötigt wird
```
- Wiederaufbau das Projekt-Loaders.
- Verknüpfen Sie Ihr Projekt.

- Es ist darauf zu achten, dass diese Änderungen beim Installieren eines neuen LASAL CLASS verloren gehen.

Die Funktionsprototypen der Comlink-Funktionen sind in der Include-Datei ComTypes.h zu finden.



Alle Funktionen mit Ausnahme der InstallCallback-Funktion können den Server blockieren und auf die Antwort des Servers warten, wenn die Schnittstelle nicht LOCAL oder INTERN ist. Diese Funktionen sollten nicht im Echtzeit- oder zyklischen Task aufgerufen werden, da sie andere Echtzeit- oder zyklische Arbeit verzögern könnten!

6.2.3 Anlegen von CAN-Identifikatoren

Die folgenden Bereiche der CAN-Identifiers sind für das Comlink-Protokoll reserviert:

- 16#700 – 16#71F
- 16#500 – 16#67F (vorausgesetzt, dass der Standardwert für die maximalen Comlink-Kanäle nicht verändert wird)

Die Identifier der CAN-Objekte, die tatsächlich angelegt werden, hängen von folgenden Faktoren ab:

- CAN-Stations-Nummer der SPS (STATION), Bereich: 0-31
- Anzahl den ausgehenden Comlink-Verbindungen (NBR_CONNECTIONS)
- Max. Anzahl der Comlink-Kanäle (MAX_CHANNELS). Der Standardwert ist 5. Dieser Wert kann angepasst werden, indem Sie im Loader (Lasal1: loader.h, Lasal2: UserDef.h) die #define COMLINK_CAN_COMCHS ändern. Nachdem dieser Wert geändert wurde, muss der Loader kompiliert und mit der Anwendung verlinkt werden. Es ist darauf zu achten, dass alle Steuerungen, die über das Comlink-Protokoll kommunizieren, den gleichen Wert verwenden!

Die folgenden Formeln braucht man zur Berechnung der zugeteilten Identifiers:

```
a) 16#700 + STATION
b) IF MAX_CHANNELS < 8 THEN
    From: 16#500 + STATION * (MAX_CHANNELS + 1) * 2
    Count: NBR_CONNECTION * 2
ELSE
    From: 16#200 + STATION * (MAX_CHANNELS + 1) * 2
    Count: NBR_CONNECTION * 2
END_IF
```

Beispiel

3 SPS mit den Stationsnummern 0, 11, 25.

Stationen 11 und 25 stellen eine Verbindung zur Comlink-Station 0 her.

Der Standardwert für COMLINK_CAN_COMCHS bleibt unverändert (-> MAX_CHANNELS = 5).

Station 0

```
STATION = 16#00, NBR_CONNECTION = 0, MAX_CHANNELS = 5
16#700 + 16#00 = 16#700
From: 16#500 + STATION * (MAX_CHANNELS + 1) * 2 = 16#500
Count: 0
```

Station 11

```
STATION = 16#0B, NBR_CONNECTION = 1, MAX_CHANNELS = 5
16#700 + 16#0B = 16#70B
From: 16#500 + STATION * (MAX_CHANNELS + 1) * 2 = 16#584
Count: 2
```

Station 25

```
STATION = 16#19, NBR_CONNECTION = 1, MAX_CHANNELS = 5
16#700 + 16#19 = 16#719
From: 16#500 + STATION * (MAX_CHANNELS + 1) * 2 = 16#62C
Count: 2
```

-> Angelegte CAN-Objekte: 16#700, 16#70B, 16#719, 16#584, 16#585, 16#62C, 16#62D

6.2.3.1 InstallCallBack

Die InstallCallBack()-Funktion installiert eine Callback-Funktion für alle bestehenden Login-Sessions. Die Callback-Funktion wird aufgerufen, wenn sich der Wert einer Update-Zelle ändert.

```
FUNCTION GLOBAL __CDECL InstallCallBack
VAR_INPUT
    pCallback : ^void;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
pCallback	^VOID	<p>Zeiger auf eine Callback-Funktion mit folgendem Prototyp:</p> <pre>FUNCTION GLOBAL __CDECL PrototypeCallback VAR_INPUT pComdef : ^COMDEF; pData : ^d2LSE; END_VAR;</pre>

Callback Funktion

Übergabeparameter	Typ	Beschreibung
pComdef	^COMDEF	Zeiger auf eine COMDEF-Struktur. Dieser Zeiger kann verwendet werden, um die Quelle der Daten zu identifizieren.
pData	^d2LSE	Zeiger auf eine d2LSE-Struktur, die folgenden Aufbau hat: d2LSE : STRUCT VarlistID : UDINT; uiOffs : UINT; Data : DINT; END_STRUCT;

Struktur d2LSE

VarlistID	UDINT	Die Zahl, die in der Funktion TxUpd für das Varlist-ID-Element des pLslcommregdata-Parameters festgelegt wurde.
UiOffs	UINT	Ein 0-basierter Index der Update-Zelle im Update-Puffer. Ein Wert von 0 bis 999 deutet darauf hin, dass die Update-Zelle aus dem statischen Update-Puffer kommt. Ein Wert größer als 1000 deutet darauf hin, dass diese aus einer dynamischen Update-Zelle kommt.
data	DINT	Daten der Update-Zelle

6.2.3.2 LDR_SetCanWait

Mit LDR_SetCanWait() wird die Zeitverzögerung zwischen zwei Übertragungen einer Comlink CAN-Meldung festgelegt. Diese Verzögerung ist notwendig, um die Überbelastung des CAN-Busses zu verhindern. Der Standardwert der Verzögerung ist 2 ms.

```
FUNCTION LDR_SetCanWait
VAR_INPUT
    us_wait : usint;
END_VAR
VAR_OUTPUT
    old : usint;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
us_wait	USINT	Verzögerungszeit in ms
Rückgabeparameter	Typ	Beschreibung

old	USINT	Bisheriger Wert der Verzögerung
-----	-------	---------------------------------

6.2.3.3 LDR_SetCanWaitRemote

Mit LDR_SetCanWaitRemote() wird die Zeitverzögerung zwischen zwei Übertragungen einer Comlink CAN-Meldung an eine abgesetzte Station festgelegt. Diese Verzögerung ist notwendig, um die Überbelastung des CAN-Busses zu verhindern. Der Standardwert der Verzögerung ist 2 ms.

```
FUNCTION LDR_SetCanWaitRemote
VAR_INPUT
    pComdef : ^COMDEF;
    wait : INT;
END_VAR
VAR_OUTPUT
    retVal : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
pComdef	^COMDEF	Zeiger auf eine COMDEF-Struktur
wait	INT	Verzögerungszeit in ms
Rückgabeparameter	Typ	Beschreibung
retVal	DINT	0 Funktion erfolgreich ≠0 Fehler-Code

6.2.3.4 LDR_SetRs232ComlinkParams

Die LDR_SetRs232ComlinkParams()-Funktion wird verwendet, um die Parameter für eine serielle Schnittstelle zur Verwendung mit einer Comlink-Verbindung einzustellen.

```
UNCTION GLOBAL __CDECL LDR_SetRs232ComlinkParams
VAR_INPUT
    interface : UDINT;
    baudrate : UINT;
    startServer : DINT;
    fnInitInterface : pVoid;
    pThis : pVoid;
END_VAR
VAR_OUTPUT
    retVal : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
Schnittstelle	UDINT	Nummer der seriellen Schnittstelle. Die Konstanten werden in der Header-Datei Comtypes.h definiert (COMLINK_IFNUM_COM1, COMLINK_IFNUM_COM2, etc.).
baudRate	UINT	Baudrate, die verwendet werden soll. Die Konstanten sind in der Header-Datei Isl_st_serial.h definiert (SERUSERBAUD_9600, SERUSERBAUD_19600 etc.). Ein Wert von 16#FFFF bedeutet, dass die Standardbaudrate im Loader definiert ist.
startServer	DINT	Ist dieses Flag auf 1 gesetzt, wird die serielle Schnittstelle vom Comlink-Server geöffnet und reagiert auf ankommende Abfragen. Ein Wert von 0 öffnet die Schnittstelle nicht. Die Verwendung der seriellen Schnittstelle vom Comlink-Server muss ausdrücklich mit einem Aufruf dieser Funktion angefordert werden. Öffnet der Server Comlink die serielle Schnittstelle, ohne angefordert zu werden, funktionieren die bestehenden Anwendungen, die die seriellen Schnittstellen verwenden, nicht.
fnInitInterface	pVoid	<p>Ein Zeiger auf eine Callback-Funktion mit folgendem Prototyp:</p> <pre>FUNCTION GLOBAL __cdecl LDR_InitRs232Interface VAR_INPUT pThis : pVoid; interfaceNbr : UDINT; hComm : pVoid; END_VAR VAR_OUTPUT // 1 : initialized // 0 : init in progress // <0 : init failed initStatus : DINT; END_VAR;</pre> <p>Wenn der Wert dieses Parameters auf eine Callback-Funktion zeigt, ruft die Comlink-Software diese Funktion auf, nachdem die serielle Schnittstelle initialisiert wurde und bevor die Kommunikation mit anderen Comlink-Knoten gestartet wird. Der Rückgabewert dieser Funktion informiert den Aufrufer, ob die Kommunikation beginnen kann. Ein Wert von 1 bedeutet, dass die Kommunikation beginnen kann. Ist die Kommunikation gestartet, wird diese Funktion in regelmäßigen Abständen aufgerufen, um der Anwendung mitzuteilen, ob sie die Kommunikation stoppen soll, weil eine Veränderung stattgefunden hat. Dann übernimmt die Funktion die Kontrolle über die serielle Schnittstelle (z.B. Aufhängen einer vorhandenen Modemverbindung). Diese Funktion wird normalerweise verwendet, um eine Dial-up-Verbindung mit einem Modem herzustellen.</p> <p>Der Parameter hComm ist ein Handle für die geöffnete serielle Schnittstelle, die als Parameter an die SERUSER API-Funktionen übertragen werden müssen.</p>

pThis	pVoid	Dieser Wert wird als Parameter pThis an die Callback-Funktion fnInitInterface übertragen. Er kann verwendet werden, um einen Objekt-Zeiger auf die globale Callback-Funktion zu übertragen.
Rückgabeparameter	Typ	Beschreibung
retval		0 Funktion erfolgreich ≠0 Fehler-Code

6.2.3.5 Login

Die Login()-Funktion versucht eine Verbindung mit dem Server herzustellen. Wenn dies erfolgreich ist, erstellt diese Funktion eine Comlink-Ressource und initialisiert eine COMDEF-Struktur, die verwendet werden kann, um auf die Ressource in anderen Funktionen zuzugreifen.

```
FUNCTION GLOBAL __CDECL Login
VAR_INPUT
  pComdef  : ^COMDEF;
END_VAR
VAR_OUTPUT
  result   : UINT;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
pComdef		^COMDEF	Zeiger auf eine COMDEF-Struktur
Rückgabeparameter		Typ	Beschreibung
result		UINT	0 Funktion erfolgreich ≠0 Fehler-Code

Ein möglicher Grund für einen Rückgabewert von 16#FFFE (Verbindung nicht hergestellt) kann sein, dass der Server nicht betriebsbereit ist. In diesem Fall kann die Login()-Funktion später wiederholt werden. Verwenden Sie die Logout()-Funktion, um die Verbindung zu beenden.

Structure COMDEF

```
Comdef : STRUCT
  Interface : UDINT;
  Address   : UINT;
  pt_COM    : ^COMDATA;
  IPAdresse : UDINT;
```

```

port      : UDINT;
res       : UDINT;
END_STRUCT;

```

Das Interface, Adressen und IP-Adressen-Element muss vom Benutzer spezifiziert werden.

Interface	UDINT	Bestimmt den Schnittstellentyp und kann einen der folgenden Werte enthalten: <ul style="list-style-type: none"> 0 LOCAL: Der Server läuft auf der gleichen Maschine wie der Client. 1 INTERN: Gleich wie LOCAL 6 CAN1: Erste CAN-Schnittstelle 8 TCP/IP1: Erste TCP/IP-Schnittstelle. Dieser Wert ist veraltet. Er wurde aus Kompatibilitätsgründen durch den Wert 10 ersetzt und ist nur für Austauschzwecke verfügbar. 10 TCP/IP1: Erste TCP/IP-Schnittstelle.
Adresse	UINT	Legt die CAN-Stationsnummer des Servers fest. Der Bereich von verfügbaren Stationsnummern reicht von 0 bis 31. Dieser Parameter ist nur gültig, wenn ein CAN-Schnittstellentyp verwendet wird.
pt_COM	^COMDATA	Die Login-Funktion speichert einen Pointer auf eine Kommunikationsstruktur für diese Comlink-Ressource ins Element pt_COM.
IPAdresse	UDINT	Legt die IP-Adresse des Servers fest. Die IP-Adresse wird in einer UDINT gespeichert, wo das High-Byte die erste Zahl der Adresse enthält. Dieser Parameter ist dann nur gültig, wenn ein TCP/IP-Schnittstelle verwendet wird.
port	UDINT	Legt die CAN-Stationsnummer des Servers fest. Dieser Parameter ist nur dann gültig, wenn eine TCP/IP-Schnittstelle mit einem Wert von ≥ 10 verwendet wird. Port-Nummer 0 bedeutet, dass der Standardwert verwendet werden soll.
res	UDINT	Ein reserviertes Element und muss auf Null gesetzt werden

6.2.3.6 StartStopRefresh

Die Funktion StartStopRefresh() teilt dem Server mit, wie viele Update-Zellen-Einträge in einem Update-Puffer gescannt werden sollen. Wenn sich der Wert einer Update-Zelle ändert, wird die Callback-Funktion aufgerufen, um den Client über den neuen Wert zu informieren.

```
FUNCTION GLOBAL __CDECL StartStopRefresh
VAR_INPUT
    pComdef : ^COMDEF;
    count   : UINT;
    typ     : UINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung				
PComdef	^COMDEF	Zeiger auf eine COMDEF-Struktur, die beim Login() initialisiert wurde				
count	UINT	Legt die Anzahl der Update-Zellen fest, die gescannt werden sollen				
typ	UINT	Legt den Typ des Update-Puffers fest. <table border="1" data-bbox="459 698 1008 777"> <tr> <td>0</td><td>Statischer Update-Puffer</td></tr> <tr> <td>1</td><td>Dynamischer Update-Puffer</td></tr> </table>	0	Statischer Update-Puffer	1	Dynamischer Update-Puffer
0	Statischer Update-Puffer					
1	Dynamischer Update-Puffer					

6.2.3.7 TxCommand

Die Funktion TxCommandEx() sendet einen Befehl an den Interpreter und wartet auf Antwort.

```
FUNCTION GLOBAL __CDECL TXCOMMAND
VAR_INPUT
    Command  : UDINT;
    length   : UDINT;
    charptr  : ^USINT;
    pComdef  : ^COMDEF;
    Presu    : ^UDINT;
END_VAR
VAR_OUTPUT
    Status   : IPRSTATES;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
Command	UDINT	Befehl an den Interpreter
length	UDINT	Länge der Daten in charptr

charptr	^USINT	Zeiger auf die Parameter des Interpreter-Befehls
pComdef	^COMDEF	Zeiger auf eine COMDEF-Struktur, die beim Login initialisiert wurde
Presu	^UDINT	Zeiger auf ein UDINT, das die Adresse des Ergebnispuffers des Interpreters empfängt
Rückgabeparameter	Typ	Beschreibung
Zustand	IPRSTATES;	Diese Funktion liefert den Zustand des Interpreters zurück, z.B.: READY, ERROR, BUSY, usw. Wenn kein Fehler auftritt, wird READY (0) zurückgegeben. Ein Rückgabewert von ERROR (1) ist nicht unbedingt ein Fehler im Interpreter des Servers. Dies kann auch auf eine fehlende Verbindung oder einen anderen Fehler im Client hinweisen. Zur Unterscheidung zwischen einem Interpreter-Fehler und anderen Fehlern, verwenden Sie die Funktion TxCommandEx() .

Diese Funktion ist für einen einzelnen Kanal nicht Thread-sicher oder Ablauf-invariant. Das bedeutet, dass es nicht zulässig ist, TxCommand() aus zwei Threads für den gleichen Kanal gleichzeitig aufzurufen. Darüber hinaus könnte bei einem Aufruf das TxCommand() in einer endlosen Schleife enden, wenn eine Lese- oder Schreibmethode über das TxCommand(Befehl I_READ oder I_WRITE) aufgerufen wird, die einen anderen TxCommand()-Anruf enthält.

Die Funktion TxCommand() darf jedoch für einen Kanal in einem Thread und für einen anderen Kanal auf einem anderen Thread aufgerufen werden.

Ist die Thread-Sicherheit erforderlich, muss die Funktion [TxCommandEx\(\)](#) verwendet werden.

6.2.3.8 TxCommandEx

Die Funktion TxCommandEx() sendet einen Befehl an den Interpreter und wartet auf Antwort. Im Vergleich zur Funktion [TxCommand\(\)](#) ist diese Funktion Thread-sicher (seit Loader-Version 1.1.69/2.2.69).

```
FUNCTION GLOBAL __CDECL TXCOMMAND
VAR_INPUT
    Command      : UDINT;
    length       : UDINT;
    charptr     : ^USINT;
    pComdef     : ^COMDEF;
    pResuBuf    : ^RESULTS;
    sizeResuBuf : UDINT;
    pReason      : ^comlinkReason;
END_VAR
VAR_OUTPUT
    Status      : IPRSTATES;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
Command		UDINT	Befehl an den Interpreter
length		UDINT	Länge der Daten in charptr
charptr		^USINT	Zeiger auf die Parameter des Interpreter-Befehls
pComdef		^COMDEF	Zeiger auf eine COMDEF-Struktur, die beim Login initialisiert wurde
pResubuf		^Results	Zeiger auf einen Puffer, der das Ergebnis des Interpreters enthält
sizeOfResuBuf		UDINT	Größe des Ergebnispuffers
pReason		^comlinkReason	Zeiger auf eine Variable, die den Ursachen-Code im Fall eines Fehlers enthält. Dieser Zeiger kann NULL sein.
Rückgabeparameter		Typ	Beschreibung
Zustand		IPRSTATES;	Diese Funktion liefert den Zustand des Interpreters zurück, z.B.: READY, ERROR, BUSY, usw. Wenn kein Fehler auftritt, wird READY (0) zurückgegeben. Ein Rückgabewert von ERROR (1) ist nicht unbedingt ein Fehler im Interpreter des Servers. Dies kann auch auf eine fehlende Verbindung oder einen anderen Fehler im Client hinweisen. Zur Unterscheidung zwischen einem Interpreter-Fehler und anderen Fehlern, verwenden Sie den pReason-Parameter, um die Ursache des Fehlers zu ermitteln. Die Werte des Ursachen-Codes sind in der Header-Datei ComTypes.h (z.B. COMLINK_ERR_NOCONNECTION) definiert.

Diese Funktion ist für einen einzelnen Kanal Thread-sicher. Das bedeutet, dass die TxCommandEx()-Methode aus zwei Threads gleichzeitig für den gleichen Kanal gerufen werden darf. Wird die Funktion TxCommandEx() während der Ausführung eines anderen Threads aufgerufen, wird der zweite Aufruf der Funktion bis zum Ende des ersten Anrufs blockiert.

6.2.3.9 TxUpd

Die Funktion TxUpd() sendet eine Update-Zelle in den Update-Puffer.

```
FUNCTION GLOBAL __CDECL TxUpd
VAR_INPUT
    pLslcommregdata : ^Lslcommregdata;
    pComdef         : ^COMDEF;
END_VAR;
```

Übergabeparameter		Typ	Beschreibung
pLslcommregdata	^Lslcommregdata		Zeiger auf eine Struktur, die die Eigenschaften der Update-Zelle definiert

PComdef	^COMDEF	Zeiger auf eine COMDEF-Struktur, die beim Login initialisiert wurde
---------	---------	---------------------------------------------------------------------

Struktur LslCommregdata

```
LslCommRegData : STRUCT
  LASALID      : UDINT;
  Channel      : UINT;
  VarPos       : UINT;
  uiTIME       : UINT;
  VarlistID    : UDINT;
END_STRUCT;
```

LASALID	UDINT	Ermittelt der LASAL-ID des Objekts fest. Die LASAL-ID erhält man mit dem Interpreter-Befehl L_GET_OBJ.
Channel	UINT	Das Kanal-Element wird von dieser Funktion nicht verwendet.
VarPos	UINT	Ermittelt den null-basierten Index der Update-Zelle in der Update-Liste.
uiTime		Das uiTime-Element bestimmt die Aktualisierungsrate der Update-Zelle. Bits von 0-14 bestimmen die Aktualisierungsrate in Millisekunden. Wird das höchstwertigste Bit (Bit 15) gesetzt, meldet das Comlink ein CRC für ein VirtualBaselinit-Objekt. Sonst meldet es das Ergebnis der Lesemethode aus dem Serverkanal. Wenn ein CRC gemeldet wird, wurde der Inhalt eines Datenpuffers verändert.
VarlistID	UDINT	Legt eine Zahl fest, die nicht interpretiert oder von der Comlink-Applikation beeinflusst wird. Kann verwendet werden, um einen vom Server gemeldeten Wert in die Kundendatenbank einzutragen.

6.2.4 Fehler-Codes

Die folgende Tabelle beschreibt die Fehler-Codes, die von den Comlink-Client-Funktionen zurückgegeben werden.

Code	Beschreibung
16#FFF0	Ein allgemein nicht wiederherstellbarer Fehler ist aufgetreten, welcher keine Informationen über die Ursache enthält.
16#FFF9	Der Server schickt eine ungültige Antwort.
16#FFFA	Die angegebene Adresse wird bereits verwendet.
16#FFFB	Die Client-Schnittsellefunktionen sind nicht vorhanden, da ein nicht wiederherstellbarer Fehler bei der Initialisierung aufgetreten ist.
16#FFFC	Die angegebene Schnittstellentyp wird nicht unterstützt.

16#FFFD	Die maximale Anzahl von Verbindungen wurde überschritten.
16#FFFE	Eine Verbindung zum Server konnte nicht hergestellt werden oder eine bestehende Verbindung wurde entfernt. Dies könnte ein temporärer Fehler sein. Die Funktion, die diesen Fehler ausgelöst hat, kann wieder aufgerufen werden.
16#FFFF	Ungültige Parameter wurden spezifiziert.

6.3 Library für SIGMATEK Hardware-Zugriff

6.3.1 Allgemein

Mit der SIGMATEK Library StkLib kann die SIGMATEK Hardware (z.B. ein IPC) mit Microsoft Systemsoftware-Anwendungen angesprochen werden. Die Programmierschnittstelle ist als DLL (Dynamic Link Library) implementiert. Je nach Art der Hardware werden Windows-Treiber benötigt.

Für die folgende Hardware sind StkLib API-Funktionen verfügbar:

- CAN Bus-Adapter BU104
- CAN-Hardware eines IPCs
- DIAS IPC-Hardware (Standard DIAS-Master)

6.3.2 Installation

Mit dem Programm DrvSetup.exe können die DLLs und StkLib Treiber installiert werden. Das Installationsprogramm bietet eine Auswahl an zu installierenden Softwarekomponenten.

Beispielprogramme für Microsoft Visual C++ werden ebenfalls im LASAL-Programmverzeichnis (normalerweise unter C:\Programme\Lasal) im Unterverzeichnis online\driver\visualC6\StkLlb installiert.

6.3.3 Anforderungen

Microsoft Windows 98, Windows ME, Windows 2000 oder Microsoft Windows XP

6.3.4 CAN APIs

Mit den CAN API-Funktionen kann auf den CAN-Controller für ein BU104 CAN Bus-Interface oder die CAN-Hardware eines IPCs zugegriffen werden. Im BU104 wird ein Intel 82526 und im IPC ein Intel 82527 CAN-Controller eingesetzt. Der Funktionsumfang des Intel 82526 ist geringer als der des Intel 82527. In der Beschreibung der API-Funktionen wird auf bestimmte Funktionen verwiesen, die für das BU104 nicht verfügbar sind.

Anzahl der verfügbaren Kommunikationsobjekte:

BU104	Die Anzahl der möglichen Kommunikationsobjekte ist abhängig von der Größe der einzelnen Objekte. Insgesamt stehen 56 Bytes für Kommunikationsobjekte zur Verfügung. Für ein Objekt werden 3 Byte zusätzlich zur Länge der Daten im CAN-Objekt benötigt.
IPC	15 Kommunikations-Objekte (eines davon ist ein CAN Basis-Objekt).

6.3.4.1 StkCanAddObject

Fügt ein CAN-Objekt in den CAN-Controller ein.

```
extern "C" int StkCanAddObject(
    HANDLE         handle,
    unsigned int   identifier,
    unsigned int   dataLengthCode,
    unsigned int   flags,
    unsigned int   localMask,
    unsigned int   rxBufQueueSize,
    BYTE           *initialData
);
```

Übergabeparameter	Typ	Beschreibung		
handle	HANDLE	CAN-Treiber-Handle		
identifier	UINT	Nachrichten-Identifier		
dataLengthCode	UINT	Länge des Nachrichten-Objekts. Die gültigen Längenwerte reichen von 0 bis 8.		
flags	UINT	Attribute des Nachrichten-Objekts. Es ist möglich, eine Kombination aus den folgenden Werten anzugeben.		
			Name	Wert
			DIR_TX	0
				Senden; ohne dieses Attribut: empfangen
			XTD_IDEN	1
			TIFIER	Das Nachrichtenobjekt verwendet einen erweiterten 29-Bit-Identifier. Ohne dieses Attribut wird ein

			Standard-11-Bit-Bezeichner verwendet. In der BU104 können keine erweiterten Bezeichner verwendet werden.
localMask	UINT	Dieser Parameter ist die lokale Maske für das 'Basic CAN Object'. „0“ bedeutet „ist egal“. „1“ bedeutet „muss übereinstimmen“. Das bedeutet, dass der Identifier der Empfangsnachricht mit dem konfigurierten Nachrichten-Identifier mit der entsprechenden Bit-Position übereinstimmen muss. Die 'Globale Maske' ist in der BU104 nicht vorhanden.	
rxBufQueueSize	UINT	Empfangene Nachrichten werden im Treiber in eine Warteschlange gestellt. Dieser Parameter zeigt die Anzahl der CAN-Nachrichten an, die in der Warteschlange des Nachrichtenobjekts gepuffert werden können.	
initialData	BYTE	Zeiger auf ein Array, das Datenbytes enthält, die in das Message-Objekt geschrieben werden. Wenn Remote-Frames verwendet werden, müssen die Daten für das Sendeobjekt initialisiert werden. Der Wert NULL zeigt an, dass die Daten des Nachrichtenobjekts mit 0 initialisiert werden.	
Übergabeparameter	Typ	Beschreibung	
		<p>>0 Positive Nummer, die zur Identifizierung des Nachrichtenobjekts im CAN-Controller verwendet werden kann.</p> <p><0 Negativer Fehler-Code</p>	

6.3.4.2 StkCanClose

Schließt ein CAN-Treiber-Handle.

```
extern "C" void StkCanClose(HANDLE handle);
```

Übergabeparameter	Typ	Beschreibung
handle	HANDLE	CAN-Treiber-Handle

6.3.4.3 StkCanOpen

Öffnet einen CAN-Treiber und gibt ein Handle zurück, das als Parameter für weitere CAN API-Funktionen übergeben wird.

```
extern "C" HANDLE StkCanOpen(
    char          *driverName,
    unsigned int   flags
);
```

Übergabeparameter	Typ	Beschreibung								
driverName	CHAR	<p>Treibername</p> <table> <tr> <td>Name</td> <td>Bedeutung</td> </tr> <tr> <td>CANLPT1</td> <td>CAN Bus-Adapter BU104 am Anschluss LPT1</td> </tr> <tr> <td>CANLPT2</td> <td>CAN Bus-Adapter BU104 am Anschluss LPT2</td> </tr> <tr> <td>CANIPC1</td> <td>CAN-Hardware des IPCs</td> </tr> </table>	Name	Bedeutung	CANLPT1	CAN Bus-Adapter BU104 am Anschluss LPT1	CANLPT2	CAN Bus-Adapter BU104 am Anschluss LPT2	CANIPC1	CAN-Hardware des IPCs
Name	Bedeutung									
CANLPT1	CAN Bus-Adapter BU104 am Anschluss LPT1									
CANLPT2	CAN Bus-Adapter BU104 am Anschluss LPT2									
CANIPC1	CAN-Hardware des IPCs									
flags	UINT	Reserviert; muss auf 0 gesetzt sein								
Übergabeparameter	Typ	Beschreibung								
		Wenn kein Fehler auftritt, wird das offene Handle an den CAN-Treiber zurückgegeben. Andernfalls wird INVALID_HANDLE_VALUE zurückgegeben.								

6.3.4.4 StkCanRxObjectAny

Diese Funktion empfängt eine CAN-Nachricht von einem beliebigen Nachrichtenobjekt im CAN-Controller.

```
extern "C" int StkCanRxObjectAny(
    HANDLE      handle,
    unsigned int *pIdentifier,
    unsigned int *pFlags,
    BYTE        *pData,
    unsigned int sizeOfData,
    unsigned int *pdataLengthCode,
    unsigned int timeout100ns
);
```

Übergabeparameter	Typ	Beschreibung
handle:	HANDLE:	CAN-Treiber-Handle
pIdentifier	UINT	Zeiger auf eine Variable, in der der Message-Identifier der empfangenen Nachricht gespeichert wird

pFlags	UINT	Zeiger auf eine Variable, in der die Attribute der empfangenen Nachricht gespeichert werden
pData :	BYTE	Zeiger auf ein Array, in dem die Daten der empfangenen Nachricht gespeichert werden
sizeOfData	UINT	Größe des Arrays pData in Byte
pDataLengthCode	UINT	Zeiger auf eine Variable, in der die Länge der Empfangsnachricht gespeichert wird. Es ist wichtig zu beachten, dass der BU104 CAN-Controller immer die konfigurierte Länge meldet und nicht die Länge der empfangenen Nachricht.
timeout100ns	UNIT	Timeout in 100 ns Einheit; 0 zeigt an, dass ein Standardwert verwendet werden soll
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich <0 Negativer Fehler-Code

6.3.4.5 StkCanRxObject

Diese Funktion empfängt eine CAN-Nachricht von einem entsprechenden Nachrichtenobjekt im CAN-Controller.

```
extern "C" int StkCanRxObject(
    HANDLE         handle,
    unsigned int   msgObjNbr,
    unsigned int   *pIdentifier,
    BYTE           *pData,
    unsigned int   sizeOfData,
    unsigned int   *pdataLengthCode,
    unsigned int   timeout100ns
);
```

Übergabeparameter	Typ	Beschreibung
handle:	HANDLE:	CAN-Treiber-Handle.
msgObjNr	UINT	Nummer des Nachrichtenobjekts im CAN-Controller (Rückgabewert von StkCanAddObject())
pIdentifier	UINT	Zeiger auf eine Variable, in der der Message-Identifier der empfangenen Nachricht gespeichert wird
pData :	BYTE	Zeiger auf ein Array, in dem die Daten der Empfangsnachricht gespeichert werden
sizeOfData	UINT	Größe des Arrays pData in Byte

pDataLengthCode	UINT	Zeiger auf eine Variable, in der die Länge der Empfangsnachricht gespeichert wird. Es ist wichtig zu beachten, dass der BU104 CAN-Controller immer die konfigurierte Länge meldet und nicht die Länge der empfangenen Nachricht.				
timeout100ns	UINT	Timeout in 100 ns Einheit; 0 zeigt an, dass ein Standardwert verwendet werden soll.				
Rückgabeparameter	Typ	Beschreibung				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td><td>Funktion erfolgreich</td></tr> <tr> <td><0</td><td>Negativer Fehler-Code</td></tr> </table>	0	Funktion erfolgreich	<0	Negativer Fehler-Code
0	Funktion erfolgreich					
<0	Negativer Fehler-Code					

6.3.4.6 StkCanSetup

Positionierung der Konfigurationsparameter des CAN-Controllers.

```
extern "C" int StkCanSetup(
    HANDLE         handle,
    unsigned int   baudRate,
    unsigned int   globalMaskStd,
    unsigned int   globalMaskExt
);
```

Übergabeparameter	Typ	Beschreibung																
handle	HANDLE	CAN-Treiber-Handle																
baudRate	UINT	CAN-Baudrate <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">0</td><td>CAN_BAUD_615KB</td></tr> <tr> <td>1</td><td>CAN_BAUD_500KB</td></tr> <tr> <td>2</td><td>CAN_BAUD_250KB</td></tr> <tr> <td>3</td><td>CAN_BAUD_125KB</td></tr> <tr> <td>4</td><td>CAN_BAUD_100KB</td></tr> <tr> <td>5</td><td>CAN_BAUD_50KB</td></tr> <tr> <td>6</td><td>CAN_BAUD_20KB</td></tr> <tr> <td>7</td><td>CAN_BAUD_1MB</td></tr> </table>	0	CAN_BAUD_615KB	1	CAN_BAUD_500KB	2	CAN_BAUD_250KB	3	CAN_BAUD_125KB	4	CAN_BAUD_100KB	5	CAN_BAUD_50KB	6	CAN_BAUD_20KB	7	CAN_BAUD_1MB
0	CAN_BAUD_615KB																	
1	CAN_BAUD_500KB																	
2	CAN_BAUD_250KB																	
3	CAN_BAUD_125KB																	
4	CAN_BAUD_100KB																	
5	CAN_BAUD_50KB																	
6	CAN_BAUD_20KB																	
7	CAN_BAUD_1MB																	
globalMaskStd	UINT	'Globale Maske' für CAN-Nachrichten mit einem 'Standard-CAN-Identifier'																

globalMaskExt	UINT	'Globale Maske' für CAN-Nachrichten mit einem 'erweiterten-CAN-Identifier'
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		<0 Negativer Fehler-Code

Die "Globale Maske" wird für die 'Nachrichtenannahmefilterung' verwendet. Aus diesem Grund können die Bits im CAN-Identifier einer Empfangsnachricht mit „ist egal“ maskiert werden. „0“ bedeutet „ist egal“. „1“ bedeutet „muss übereinstimmen“. Das bedeutet, dass der Identifier der Empfangsnachricht mit dem konfigurierten Nachrichten-Identifier mit der entsprechenden Bit-Position übereinstimmen muss. Die 'Globale Maske' ist in der BU104 nicht vorhanden.

6.3.4.7 StkCanSwReset

Löst einen Software-Reset im CAN-Controller aus.

```
extern "C" int StkCanSwReset(HANDLE handle);
```

Übergabeparameter	Typ	Beschreibung
handle	HANDLE	CAN-Treiber-Handle
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		<0 Negativer Fehler-Code

Befindet sich der CAN-Controller im Zustand BUS-OFF, muss er über die Software zurückgesetzt werden. Der Software-Reset für die CAN-Hardware im IPC muss nicht durch das Applikationsprogramm ausgeführt werden. In diesem Fall führt der Treiber den Software-Reset automatisch aus. Ein Rückgabewert von 1004/CANERR_DRV_R_BUS_STATUS bedeutet, dass der Status des CAN-Controllers im Zustand BUS-OFF ist.

6.3.4.8 StkCanTxObject

Versendet eine CAN-Nachricht.

```
extern "C" int StkCanTxObject(
    HANDLE         handle,
    unsigned int   msgObjNbr,
    unsigned int   identifier,
    unsigned int   dataLengthCode,
    unsigned int   flags,
    BYTE          *pData,
    unsigned int   timeout100ns
);
```

Übergabeparameter	Typ	Beschreibung		
handle	HANDLE	CAN-Treiber-Handle		
msgObjNr	UINT	Nummer des Nachrichtenobjekts im CAN-Controller (Rückgabewert von StkCanAddObject())		
identifier	UINT	Nachrichten-Identifier		
dataLengthCode	UINT	Länge der Nachricht; die gültigen Längenwerte reichen von 0 bis 8		
flags	UINT	Attribute des Nachrichten-Objekts Name Wert Bedeutung: XTD_IDE NTIFIER 1 Das Nachrichtenobjekt verwendet einen erweiterten 29-Bit-Identifier. Ohne dieses Attribut wird ein Standard-11-Bit- Bezeichner verwendet. In der BU104 können keine erweiterten Bezeichner verwendet werden.		
pData	BYTE	Zeiger auf ein Array, das die zu sendenden Daten enthält		
Timeout100ns	UINT	Timeout in 100 ns Einheit; 0 zeigt an, dass ein Standardwert verwendet werden soll		
		Rückgabeparameter	Typ	Beschreibung
				0 Funktion erfolgreich
				<0 Negativer Fehler-Code

6.3.5 DIAS APIs

Die DIAS-API-Funktionen ermöglichen den Zugriff auf die DIAS-Hardware (Standard-DIAS-Master) in einem IPC. Es ist nicht möglich, auf DIAS-Master-Interrupts zu reagieren.

Es können bis zu 64 DIAS-Module an den DIAS-Bus angeschlossen werden. Dem DIAS-Modul wird eine Adresse in einem Wertebereich von 0-63 zugewiesen. Das DIAS-Modul hat einen Adressbereich von 256 Datenbytes und 256 Steuerregisterbytes. Die Bedeutung dieser Bytes ist abhängig vom DIAS-Modultyp. Zum Beispiel können die Ausgänge 0-15 eines digitalen Ausgangsmoduls DTO163 adressiert werden, indem das Wort an die Datenadresse 0 geschrieben wird. Die Modulkennung für ein DIAS-Modul finden Sie im Steuerregister-Byte an Adresse 255. Weitere Informationen zu den einsetzbaren DIAS-Modultypen finden Sie in der DIAS-Dokumentation.

Die modulunabhängigen Steuerregister des DIAS-Masters befinden sich in einem 16-kByte-Adressbereich.

6.3.5.1 StkDiasCheckError

Liefert den Inhalt des DIAS-Fehlerregisters.

```
extern "C" int StkDiasCheckError(
    HANDLE handle
);
```

Übergabeparameter		Typ	Beschreibung
handle		HANDLE	CAN-Treiber-Handle
Rückgabeparameter		Typ	Beschreibung
			0 Funktion erfolgreich <0 Negativer Fehler-Code

6.3.5.2 StkDiasClose

Schließt ein CAN-Treiber-Handle.

```
extern "C" void StkDiasClose(HANDLE handle);
```

Übergabeparameter		Typ	Beschreibung
handle		HANDLE	CAN-Treiber-Handle

6.3.5.3 StkDiasGetModID

Liefert die DIAS-ID eines Moduls an eine angegebene Adresse.

```
extern "C" int StkDiasGetModID(
    HANDLE handle,
    BYTE mod,
    BYTE *id
);
```

Übergabeparameter	Typ	Beschreibung	
handle	HANDLE	CAN-Treiber-Handle	
MOD	BYTE	Moduladresse (0-63).	
id	BYTE	Zeiger auf eine Variable, in der die DIAS-ID gespeichert wird. Befindet sich kein Modul an dieser Adresse, wird der Wert 255 zurückgegeben.	
Rückgabeparameter	Typ	Beschreibung	
		0 Funktion erfolgreich <0 Negativer Fehler-Code	

6.3.5.4 StkDiasOpen

Öffnet einen DIAS-Treiber und gibt ein Handle als Parameter für weitere DIAS-API-Funktionen zurück.

```
extern "C" HANDLE StkDiasOpen(
    char *driverName,
    unsigned int flags
);
```

Übergabeparameter	Typ	Beschreibung	
driverName	CHAR	Treibername	
		Name	Bedeutung
		CANIPC1	DIAS IPC-Hardware (Standard DIAS-Master)
flags	UINT	Reserviert; muss auf 0 gesetzt werden	
Rückgabeparameter	Typ	Beschreibung	
		0 Funktion erfolgreich	

		<0 Negativer Fehler-Code
--	--	--------------------------

6.3.5.5 StkDiasReadByte

Liest ein Byte aus dem Datenbereich eines DIAS-Moduls.

```
extern "C" int StkDiasReadByte(
    HANDLE handle,
    BYTE mod,
    WORD ofs,
    BYTE *pData
);
```

Übergabeparameter	Type	Beschreibung
handle	HANDLE	CAN-Treiber-Handle
mod	BYTE	Moduladresse (0-63)
ofs	WORD	Offset im Modul-Adressbereich
pData	BYTE	Zeiger auf eine Variable, in der das gelesene Byte gespeichert wird
Rückgabeparameter	Type	Beschreibung
		0 Funktion erfolgreich
		<0 Negativer Fehler-Code

6.3.5.6 StkDiasReadCtrl

Liest das Steuerregister-Byte eines DIAS-Moduls.

```
extern "C" int StkDiasReadCtrl(
    HANDLE handle,
    BYTE mod,
    WORD ofs,
    BYTE *pData
);
```

Übergabeparameter	Typ	Beschreibung
handle	HANDLE	CAN-Treiber-Handle
MOD	BYTE	Moduladresse (0-63)
ofs	WORD	Offset im Modul-Adressbereich

pData	BYTE	Zeiger auf eine Variable, in der das gelesene Byte gespeichert wird
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		<0 Negativer Fehler-Code

6.3.5.7 StkDiasReadWord

Liest ein Wort aus dem Datenbereich eines geeigneten DIAS-Moduls.

```
extern "C" int StkDiasReadWord(
    HANDLE handle,
    BYTE mod,
    WORD ofs,
    WORD *pData
);
```

Übergabeparameter	Typ	Beschreibung
handle	HANDLE	CAN-Treiber-Handle
MOD	BYTE	Moduladresse (0-63)
ofs	WORD	Offset im Modul-Adressbereich
pData	WORD	Zeiger auf eine Variable, in der das gelesene Wort gespeichert wird
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		<0 Negativer Fehler-Code

6.3.5.8 StkDiasReadRegister

Liest das Steuerregister-Byte aus dem DIAS-Adressbereich.

```
extern "C" int StkDiasReadRegister(
    HANDLE handle,
    WORD ofs,
    BYTE *pData
);
```

Übergabeparameter	Typ	Beschreibung
handle	HANDLE	CAN-Treiber-Handle

ofs	WORT	Register-Offset im DIAS-Adressbereich
pData	BYTE*	Zeiger auf eine Variable, in der das gelesene Byte gespeichert wird
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich <0 Negativer Fehler-Code

6.3.5.9 StkDiasWriteByte

Schreibt ein Byte in den Datenbereich eines entsprechenden DIAS-Moduls.

```
extern "C" int StkDiasWriteByte(
    HANDLE handle,
    BYTE mod,
    WORD ofs,
    BYTE data
);
```

Übergabeparameter	Typ	Beschreibung
handle	HANDLE	CAN-Treiber-Handle
mod	BYTE	Moduladresse (0-63)
ofs	WORD	Offset im Modul-Adressbereich
data	BYTE	Byte, das geschrieben werden soll
Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich <0 Negativer Fehler-Code

6.3.5.10 StkDiasWriteCtrl

Liest das Steuerregister-Byte eines DIAS-Moduls.

```
extern "C" int StkDiasWriteCtrl(
    HANDLE handle,
    BYTE mod,
    WORD ofs,
    BYTE data
);
```

Übergabeparameter	Typ	Beschreibung	
handle	HANDLE	CAN-Treiber-Handle	
mod	BYTE	Moduladresse (0-63)	
ofs	WORD	Offset im Modul-Adressbereich	
data	BYTE	Byte, das geschrieben werden soll	
Rückgabeparameter	Typ	Beschreibung	
		0	Funktion erfolgreich
		<0	Negativer Fehler-Code

6.3.5.11 StkDiasWriteRegister

Liest das Steuerregister-Byte aus dem DIAS-Adressbereich.

```
extern "C" int StkDiasWriteRegister(
    HANDLE handle,
    WORD    ofs,
    BYTE    data
);
```

Übergabeparameter	Typ	Beschreibung	
handle	HANDLE	CAN-Treiber-Handle	
ofs	WORD	Register-Offset im DIAS-Adressbereich	
data	BYTE	Byte, das geschrieben werden soll	
Rückgabeparameter	Typ	Beschreibung	
		0	Funktion erfolgreich
		<0	Negativer Fehler-Code

6.3.5.12 StkDiasWriteWord

Schreibt ein Wort in den Datenbereich eines DIAS-Moduls.

```
extern "C" int StkDiasWriteWord(
    HANDLE handle,
    BYTE    mod,
    WORD    ofs,
    WORD    wData
);
```

Übergabeparameter		Typ	Beschreibung
handle	HANDLE		CAN-Treiber-Handle
MOD	BYTE		Moduladresse (0-63)
ofs	WORT		Offset im Modul-Adressbereich
wData	WORT		Wort, das geschrieben werden soll
Rückgabeparameter		Typ	Beschreibung
			0 Funktion erfolgreich
			<0 Negativer Fehler-Code

6.4 API Safety DLL

6.4.1 Allgemeine Informationen

Dieses Dokument enthält eine Beschreibung der Funktionen aus dem „Safety DLL Interface“. Diese Funktionen werden verwendet, um von einer Standard-SPS aus eine Kommunikationsverbindung zu einer Safe-CPU aufzubauen und folgende Abläufe auszuführen:

- Download einer Konfiguration
- Setzen des Verified-Flags
- Ändern des Passworts
- Starten der Safe-CPU (Verlassen des Service-Modus)
- Stoppen der Safe-CPU (Herstellen des Service-Modus)
- Statusabfrage
- Neustart der Safety-Applikation
- Quittieren eines Fehlers



Die Funktionen dürfen nicht dafür verwendet werden, um vollautomatisch einen Ablauf durchzuführen. Es muss vom Programmhersteller sichergestellt werden, dass die Eingabedaten vom Benutzer stammen und nicht im Programm hardcodiert werden.

Wenn Funktionen, die den File-State oder den Connection-State verändern sollen, fehlschlagen, bleibt der File-State und der Connection-State unverändert.

6.4.2 Interface-Funktion ISAFETY_DLL

Die Funktionen stehen beim LasalOS erst dann zur Verfügung, wenn die Safety-DLL geladen wurde. Dazu muss sich die Datei safety.dlm im Verzeichnis C:\LSLSYS befinden und der CLI-Befehl loadsafety ausgeführt werden. Beim Salamander-Betriebssystem ist dieser Schritt nicht notwendig.

Um die Schnittstelle zu verwenden, muss die Header-Datei lsl_st_safety.dll.h eingebunden und ein Zeiger über OS_CILGET auf die Schnittstellenstruktur geholt werden. Dieser Zeiger wird als erster Parameter bei den Makros der Schnittstellenfunktionen angegeben.

Der Name der Schnittstelle ist ISAFETY_DLL und ist mit dem Makro INTERFACE_SAFETY_DLL definiert.

Beispiel

```
#include <lsl_st_safetydll.h>

VAR_GLOBAL
  pSafety : ^OS_SAFETY_DLL;
END_VAR

FUNCTION VIRTUAL GLOBAL Class0::Init
  IF _FirstScan THEN
    OS_CILGet(INTERFACE_SAFETY_DLL, #pSafety$void);
    SAFETY_NEW_STATE(pSafety);
  END_IF;
END_FUNCTION
```

Die Funktionen sind nicht Thread-Safe, d.h. sie dürfen nicht aus mehreren verschiedenen Threads aus aufgerufen werden.

Manche der Funktionen sind blockierend, d.h. es kann eine gewisse Zeit dauern, bis die Funktion ausgeführt wurde. Diese Funktionen sollten daher nicht in einer zyklischen Funktion eines LASAL-Objekts (background, cywork, rtwork) aufgerufen werden, weil dadurch die Abarbeitung der übrigen zyklischen Funktion angehalten werden würde.

Stattdessen sollte der Aufruf in einem eigenen Thread erfolgen (siehe MultiTask Interface).

Beispiel

```

FUNCTION SafetyThread
VAR_INPUT
    thisPointer : pVoid;
END_VAR
/* TODO: insert ISAFETY_DLL functions */
END_FUNCTION

/* toMT is an object-channel to _MultiTask */
toMT.CREATETHREAD(#SafetyThread(), 10, 8192, 0, this, "Safety");

```

6.4.2.1 SAFETY_CHANGE_PASSWORD

Ändert das Passwort auf der Ziel-Safe-CPU.

Vor dem Aufruf dieser Funktion muss [SetUserPromptTime\(\)](#) aufgerufen und vom Benutzer eine Bestätigung über die erfolgreiche Übertragung angefordert werden. Nach dem Aufruf von [SetUserPromptTime\(\)](#) muss der Aufruf dieser Funktion innerhalb von 60 Sekunden stattfinden.

```

FUNCTION __CDECL GLOBAL P_Safety_ChangePassword
VAR_INPUT
    Level      : USINT;
    oldPassword : ^CHAR;
    newPassword : ^CHAR;
END_VAR
VAR_OUTPUT
    retval     : DINT;
END_VAR;

#define SAFETY_CHANGE_PASSWORD(pCil,p1,p2,p3)
    pCil^.ChangePassword
    $ P_Safety_ChangePassword(p1,p2,p3)

```

Übergabeparameter		Typ	Beschreibung
level	USINT	Login level 1 Debug level 2 Configuration level	
oldPassword	^CHAR	Altes Passwort Größe: 8 Bytes. wenn das Passwort kleiner als 8 Zeichen ist, muss der Rest mit Leerzeichen aufgefüllt werden.	
newPassword	^CHAR	Neues Passwort Größe 8 Byte. Wenn das Passwort kleiner als 8 Zeichen ist, muss der Rest mit Leerzeichen aufgefüllt werden.	
Rückgabeparameter		Typ	Beschreibung

retval	DINT	0 Funktion erfolgreich ≠0 Fehler-Code
--------	------	------------------------------------------

Anforderungen

Connection-State \geq CS_CONNECTION und File-State \geq FS_FILE.

6.4.2.2 SAFETY_CHECK_DONGLE_FW

Überprüft, ob unter der Sicherheitsnummer ein Dongle eingesteckt ist.

Setzt den File-State auf FW_DONGLE, wenn das Modul im Dongle-Modus ist.

```
FUNCTION __CDECL GLOBAL P_Check_Dongle_FW
VAR_INPUT
    safetyNbr : UDINT;
END_VAR
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_CHECK_DONGLE_FW(pCil,p1)
    pCil^.CheckDongleFW
    $ P_Check_Dongle_FW(p1)
```

Übergabeparameter	Typ	Beschreibung	
safetyNbr	UDINT	Sicherheitsnummer des Moduls mit Dongle	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0 Funktion erfolgreich ≠0 Fehler-Code	

Anforderungen

File-State == FW_FILE

6.4.2.3 SAFETY_CLEAR_CONFIG

Löscht die Konfiguration in den verbundenen Modulen.

```
FUNCTION __CDECL GLOBAL P_Safety_ClearConfig
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_CLEAR_CONFIG(pCil)
    pCil^.ClearConfig
    $ P_Safety_ClearConfig()
```

Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich
		#0 Fehler-Code

Anforderungen

Connection-State == CS_CONNECTION

6.4.2.4 SAFETY_DELETE_STATE

Entfernt den Safety-State. Muss am Ende aufgerufen werden. Eine aufrechte Kommunikationsverbindung zur Safe-CPU wird geschlossen.

```
FUNCTION __CDECL GLOBAL P_Safety_DeleteState
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_DELETE_STATE(pCil)
    pCil^.DeleteState
    $ P_Safety_DeleteState()
```

Rückgabeparameter	Typ	Beschreibung
		0 Funktion erfolgreich
		#0 Fehler-Code

6.4.2.5 SAFETY_DOWNLOAD_FILE

Überträgt einen Teil des zuvor angegebenen Download-Files zur Safe-CPU. Vor dem ersten Aufruf muss die Anzahl der bereits übertragenen Byte (pBytesTransferred) auf 0 gesetzt werden. Die Übertragung ist beendet, sobald die Anzahl der bereits übertragenen Byte der Anzahl der insgesamt zu übertragenden Bytes (pTransferSize) entspricht.

```
FUNCTION __CDECL GLOBAL P_Safety_DownloadFile
```

```
VAR_INPUT
```

```
    pBytesTransferred : ^UDINT;
```

```
    pTransferSize      : ^UDINT;
```

```
END_VAR
```

```
VAR_OUTPUT
```

```
    retval            : DINT;
```

```
END_VAR;
```

```
#define SAFETY_DOWNLOAD_FILE(pCil,p1,p2)
```

```
    pCil^.DownloadFile
```

```
    $ P_Safety_DownloadFile(p1,p2)
```

Übergabeparameter	Typ	Beschreibung	
pBytesTransferred : ^UDINT;	^UDINT	Zeiger auf die Variable in der die bereits übertragenen Byte gespeichert sind. Der Wert dieser Variablen muss vor dem ersten Aufruf auf 0 gesetzt werden. Nach dem Aufruf dieser Funktion ist in diesen Variablen die Anzahl der bisher übertragenen Byte gespeichert.	
pTransferSize	^UDINT	Zeiger auf die Variable, in die die Anzahl der insgesamt zu übertragenden Byte geschrieben wird	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0 Funktion erfolgreich ≠0 Fehler-Code	

Anforderungen

Connection-State == CS_LOGGED_IN und File-State == FS_FILE.

6.4.2.6 SAFETY_DOWNLOAD_FW

Überträgt einen Teil des zuvor angegebenen Image-Files zur Safe-CPU. Vor dem ersten Aufruf muss die Anzahl der bereits übertragenen Byte (pBytesTransferred) auf 0 gesetzt werden. Die Übertragung ist beendet, sobald die Anzahl der bereits übertragenen Byte der Anzahl der insgesamt zu übertragenden Bytes (pTransferSize) entspricht.

```
FUNCTION __CDECL GLOBAL P_Download_FW
```

```

VAR_INPUT
    pBytesTransferred : ^UDINT;
    pTransferSize     : ^UDINT;
END_VAR
VAR_OUTPUT
    retval           : DINT;
END_VAR;

#define SAFETY_DOWNLOAD_FW(pCil,p1,p2)
    pCil^.DownloadFW
    $ P_Download_FW(p1,p2)

```

Übergabeparameter	Typ	Beschreibung
pBytesTransferred		Zeiger auf die Variable in der die bereits übertragenen Byte gespeichert sind. Der Wert dieser Variablen muss vor dem ersten Aufruf auf 0 gesetzt werden. Nach dem Aufruf dieser Funktion ist in diesen Variablen die Anzahl der bisher übertragenen Byte gespeichert.
pTransferSize		Zeiger auf die Variable, in die die Anzahl der insgesamt zu übertragenden Byte geschrieben wird
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich ≠0 Fehler-Code

Anforderungen

Connection-State == CS_LOGGED_IN und File-State == FW_DONGLE

6.4.2.7 SAFETY_FILE_GET_PRJNAME

Liefert den in der Download-Datei gespeicherten Projektnamen.

```

FUNCTION __CDECL GLOBAL P_Safety_FileGetPrjName
VAR_INPUT
    prjName       : ^CHAR;
    bufsizePrjName : UDINT;
END_VAR
VAR_OUTPUT
    retval       : DINT;
END_VAR;

#define SAFETY_FILE_GET_PRJNAME(pCil,p1,p2)
    pCil^.FileGetPrjName
    $ P_Safety_FileGetPrjName(p1,p2)

```

Übergabeparameter		Type	Beschreibung	
prjName		^CHAR	Puffer, in den der Projektname als 0-terminierter String geschrieben wird	
bufSize_prjName		UDINT	Puffergröße	
Rückgabeparameter		Type	Beschreibung	
retval		DINT	0	Funktion erfolgreich
			#0	Fehler-Code

Anforderungen

File-State \geq FS_FILE

6.4.2.8 SAFETY_FILE_GET_REVNBR

Liefert die in der Download-Datei gespeicherte Revisionsnummer.

```
FUNCTION __cdecl GLOBAL P_Safety_FileGetRevNbr
VAR_INPUT
    revNbr      : ^CHAR;
    bufsizeRevNbr : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;

#define SAFETY_FILE_GET_REVNBR(pCil,p1,p2)
    pCil^.FileGetRevNbr
    $ P_Safety_FileGetRevNbr(p1,p2)
```

Übergabeparameter		Typ	Beschreibung	
revNbr		^CHAR	Puffer, in den die Revisionsnummer als 0-terminierter String geschrieben wird	
bufSize_revNbr		UDINT	Puffergröße	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	0	Funktion erfolgreich
			#0	Fehler-Code

Anforderungen

File-State ≥ FS_FILE

6.4.2.9 SAFETY_FILE_GET_SCPUNAME

Liefert den in der Download-Datei gespeicherten SCPU-Namen.

```
FUNCTION __cdecl GLOBAL P_Safety_FileGetScpuName
VAR_INPUT
    scpuName      : ^CHAR;
    bufSizeScpuName : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;

#define SAFETY_FILE_GET_SCPUNAME(pCil,p1,p2)
    pCil^.FileGetScpuName
    $ P_Safety_FileGetScpuName(p1,p2)
```

Übergabeparameter		Typ	Beschreibung	
scpuName		^CHAR	Puffer, in den der SCPU-Name als 0-terminierter String geschrieben wird	
bufSize_scpuName		UDINT	Puffergröße	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	0 Funktion erfolgreich ≠0 Fehler-Code	

Anforderungen

File-State ≥ FS_FILE.

6.4.2.10 SAFETY_GET_CFGSTATE

Liefert den Konfigurations-Status der Safe-CPU.

```
FUNCTION __cdecl GLOBAL P_Safety_GetCfgState
VAR_INPUT
    pCfgState : ^USINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
```

```
END_VAR;

#define SAFETY_GET_CFGSTATE(pCil,p1)
  pCil^.GetCfgState
  $ P_Safety_GetCfgState(p1)
```

Übergabeparameter	Typ	Beschreibung	
pCfgState	^USINT	Zeiger auf eine Variable, in die der Konfigurations-Status geschrieben wird	
		1 INVALID 2 NOT_CONFIGURED 4 CONFIGURED_NOT_DEPLOYED_NOT_VERIFIED 8 CONFIGURED_AND_VERIFIED 16 CONFIGURED_DEPLOYED_NOT_VERIFIED 36 CONFIGURED_NOT_DEPLOYED_NOT_VERIFIED_DEV 48 CONFIGURED_DEPLOYED_NOT_VERIFIED_DEV	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0 Funktion erfolgreich ≠0 Fehler-Code	

Anforderungen

Connection-State ≥ CS_CONNECTION.

6.4.2.11 SAFETY_GET_IMAGE_MOD_ID_FW

Liefert die in der Image-Datei gespeicherte Modul-ID.

```
FUNCTION __CDECL GLOBAL P_Get_Image_Modul_ID_Fw
VAR_INPUT
  pModID : ^UDINT;
END_VAR
VAR_OUTPUT
  retval : DINT;
END_VAR;

#define SAFETY_GET_IMAGE_MOD_ID_FW(pCil,p1)
  pCil^.GetImageModulIdFw
  $ P_Get_Image_Modul_ID_Fw(p1)
```

Übergabeparameter		Typ	Beschreibung	
pModID		^UDINT	Zeiger auf den Puffer für den Modul ID (4 Byte)	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	0 Funktion erfolgreich ≠0 Fehler-Code	

Anforderungen

File-State == FW_FILE

6.4.2.12 SAFETY_GET_IMAGE_VERSION_FW

Liefert die in der Image-Datei gespeicherte (Minor-) Version.

```
FUNCTION __CDECL GLOBAL P_Get_Image_Version_Fw
VAR_INPUT
  pVersion : ^UDINT;
END_VAR
VAR_OUTPUT
  retval : DINT;
END_VAR;

#define SAFETY_GET_IMAGE_VERSION_FW(pCil,p1)
  pCil^.GetImageVersionFW
  $ P_Get_Image_Version_Fw(p1)
```

Übergabeparameter		Typ	Beschreibung	
pVersion		^UDINT	Zeiger auf den Puffer für den Versions-Nummer (4 Byte)	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	0 Funktion erfolgreich ≠0 Fehler-Code	

Anforderungen

File-State == FW_FILE

6.4.2.13 SAFETY_GET_MODUL_VERSION_FW

Liefert die (Minor-) Version des Moduls, für die die Sicherheitsnummer angegeben wurde.

```
FUNCTION __CDECL GLOBAL P_Get_Modul_Version_Fw
VAR_INPUT
    pVersion : ^SAFETY_MODUL_VERSION;
END_VAR
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_GET_MODUL_VERSION_FW(pCil,p1)
    pCil^.GetModulVersionFW
    $ P_Get_Modul_Version_Fw(p1)
```

Transfer parameters	Typ	Beschreibung	
pVersion	^SAFETY_MODUL_VERSION	Zeiger auf den Buffer für die Versionsnummer (Struktur vom Typ SAFETY_MODUL_VERSION)	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	Funktion erfolgreich
		≠0	Fehler-Code

Anforderungen

Connection-State == CS_CONNECTION

6.4.2.14 SAFETY_GET_RUNSTATE

Liefert den Run-Status der Safe-CPU.

```
FUNCTION __CDECL GLOBAL P_Safety_GetRunState
VAR_INPUT
    pRunState : ^USINT;
END_VAR
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_GET_RUNSTATE(pCil,p1)
    pCil^.GetRunState
    $ P_Safety_GetRunState(p1)
```

Übergabeparameter	Typ	Beschreibung
-------------------	-----	--------------

pRunState	^USINT	Zeiger auf eine Variable, in die der Run-Status geschrieben wird														
		<table border="1"> <tr><td>1</td><td>POST</td></tr> <tr><td>2</td><td>DIENST</td></tr> <tr><td>3</td><td>Error</td></tr> <tr><td>8</td><td>IDLE</td></tr> <tr><td>16</td><td>CHK_CFG</td></tr> <tr><td>32</td><td>OP_TEMP</td></tr> <tr><td>64</td><td>OP</td></tr> </table>	1	POST	2	DIENST	3	Error	8	IDLE	16	CHK_CFG	32	OP_TEMP	64	OP
1	POST															
2	DIENST															
3	Error															
8	IDLE															
16	CHK_CFG															
32	OP_TEMP															
64	OP															
Rückgabeparameter	Typ	Beschreibung														
retval	DINT	<table border="1"> <tr><td>0</td><td>Funktion erfolgreich</td></tr> <tr><td>#0</td><td>Fehler-Code</td></tr> </table>	0	Funktion erfolgreich	#0	Fehler-Code										
0	Funktion erfolgreich															
#0	Fehler-Code															

Anforderungen

Connection-State \geq CS_CONNECTION.

6.4.2.15 SAFETY_GET_SAFETY_NBR

Stellt eine Kommunikationsverbindung zu der in der Datei gespeicherten Safe-CPU her und fragt die Sicherheitsnummer ab.

```
FUNCTION __cdecl GLOBAL P_Safety_GetSafetyNbr
VAR_INPUT
    pSafetyNbr : ^UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;

#define SAFETY_GET_SAFETY_NBR(pCil,p1)
    pCil^.GetSafetyNbr
    $ P_Safety_GetSafetyNbr(p1)
```

Übergabeparameter	Typ	Beschreibung
pSafetyNbr	^UDINT	Zeiger auf die Variable, in der die abgefragte Sicherheitsnummer geschrieben wird
Rückgabeparameter	Typ	Beschreibung
retval	DINT	

retval	DINT	0 Funktion erfolgreich ≠0 Fehler-Code
--------	------	------------------------------------------

Anforderungen

File-State \geq FS_FILE.

6.4.2.16 SAFETY_LEAVE_SERVICE_MODE

Weist die Safe-CPU an, den Run-Status SERVICE zu verlassen.

```
FUNCTION __CDECL GLOBAL P_Safety_LeaveServiceMode
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_LEAVE_SERVICE_MODE(pCil)
  pCil^.LeaveServiceMode
  $ P_Safety_LeaveServiceMode()
```

Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich ≠0 Fehler-Code

Anforderungen

Connection-State == CS_LOGGED_IN.

6.4.2.17 SAFETY_LOGIN

Ein- oder Ausloggen auf der Ziel-Safe-CPU.

Vor dem Aufruf dieser Funktion muss [SetUserPromptTime\(\)](#) aufgerufen und das Passwort vom Benutzer angefordert werden. Nach dem Aufruf von [SetUserPromptTime\(\)](#) muss der Aufruf dieser Funktion innerhalb von 60 Sekunden stattfinden.

Sets Connection-State auf CS_LOGGED_IN

```
FUNCTION __CDECL GLOBAL P_Safety_Login
VAR_INPUT
    Level      : USINT;
    password  : ^CHAR;
END_VAR
```

```

VAR_OUTPUT
  retval    : DINT;
END_VAR;

#define SAFETY_LOGIN(pCil,p1,p2)
  pCil^.Login
  $ P_Safety_Login(p1,p2)

```

Übergabeparameter		Typ	Beschreibung
level	USINT	Login-Level	
		0	Logout
		1	Debug-Level
		2	Configuration-Level
Rückgabeparameter		Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich ≠0 Fehler-Code	

Anforderungen

Connection-State == CS_CONNECTION.

6.4.2.18 SAFETY_NEW_STATE

Erstellt einen neuen Safety-State, der für den Aufruf der nachfolgenden Funktionen erforderlich ist. Muss am Anfang einmal aufgerufen werden. Der Safety-State bleibt so lange erhalten, bis er durch SAFETY_DELETE_STATE entfernt wird.

Im Safety-State werden folgende Elemente gespeichert:

Connection-State	Zustand der Download-Datei	
	FS_IDLE	wenn noch keine Download-Datei angegeben wurde
	FS_FILE	wenn eine Download-Datei angegeben wurde und gelesen werden konnte
	FS_DOWNLOAD	wenn eine Download-Datei zur Safe-CPU übertragen wurde
Zustand der Kommunikationsverbindung zur Safe-CPU		
		CS_IDLE wenn noch keine Sicherheitsnummer angegeben wurde

	CS_SAFETY_NBR	wenn eine Sicherheitsnummer angegeben wurde
	CS_CONNECTION	wenn eine Verbindung aufgebaut wurde
	CS_LOGGED_IN	wenn ein Login durchgeführt wurde (Developer-Login)
User Prompt Time		Zeitpunkt des Aufrufs von SetUserPromptTime()
Project Name		Name des in der Download-Datei angegebenen Projekts
Revisionsnummer		Revisions-Nummer des in der Download-Datei angegebenen Projekts
S-CPU Name		Name der in der Download-Datei angegebenen Ziel-Safe-CPU
HW Path		Hardware-Pfad zur Ziel-Safe-CPU

Setzt den Datei-Status auf FS_IDLE und den Connection-State auf CS_IDLE.

```
FUNCTION __CDECL GLOBAL P_Safety_NewState
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_NEW_STATE(pCil)
    pCil^.NewState
    $ P_Safety_NewState()
```

Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	Funktion erfolgreich

6.4.2.19 SAFETY_OPEN_CONNECTION

Stellt eine Kommunikationsverbindung zur Ziel-Safe-CPU mit der zuvor angegebenen Sicherheitsnummer her. Wenn eine Download-Datei angegeben wurde, dann wird für die nicht sichere Netzwerkadresse der Hardware-Pfad aus der Download-Datei verwendet. Andernfalls werden alle angeschlossenen Safe-CPUs aufgelistet und diejenige mit der passenden Sicherheitsnummer herausgesucht.

Zu beachten ist, dass bei einer aufrechten Kommunikationsverbindung regelmäßig eine Kommunikation stattfinden muss, sonst wird diese nach einer Timeout-Zeit von 10 Sekunden von der Safe-CPU geschlossen.

Setzt Connection-State auf CONNECTION

```
FUNCTION __CDECL GLOBAL P_Safety_OpenConnection
VAR_OUTPUT
    retval : DINT;
```

```
END_VAR;

#define SAFETY_OPEN_CONNECTION(pCil)
  pCil^.OpenConnection
  $ P_Safety_OpenConnection()
```

Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich #0 Fehler-Code

Anforderungen

Connection-State == SAFETY_NBR.

6.4.2.20 SAFETY_QUIT_ERROR

Schickt einen Quit-Error Befehl an die Safe-CPU.

```
FUNCTION __cdecl GLOBAL P_Safety_QuitError
VAR_INPUT
  quitRemoteModules : USINT;
END_VAR
VAR_OUTPUT
  retval : DINT;
END_VAR;

#define SAFETY_QUIT_ERROR(pCil,p1)
  pCil^.QuitError
  $ P_Safety_QuitError(p1)
```

Übergabeparameter	Typ	Beschreibung
quitRemoteModules	USINT	Flag, das anzeigt, ob nur der Fehler im angesprochenen Modul quittiert werden soll oder auch in allen entfernten Modulen 0 es wird nur der Fehler im angesprochenen Modul quittiert #0 die Safe-CPU schickt zusätzlich an die entfernten Module einen Quit-Error Befehl
		Dieser Parameter wird in der Safe-CPU erst ab Version 337 berücksichtigt
Rückgabeparameter	Typ	Beschreibung

retval	DINT	0 Funktion erfolgreich
		#0 Fehler-Code

Anforderungen

Connection-State \geq CS_CONNECTION

6.4.2.21 SAFETY_SET_CONFIGURED

Bestätigt eine erfolgreiche Übertragung des Download-Files, indem der Konfigurations-Status in der Safe-CPU auf „konfiguriert + nicht-verifiziert + nicht-verteilt“ gesetzt wird.

Vor dem Aufruf dieser Funktion muss [SetUserPromptTime\(\)](#) aufgerufen und vom Benutzer eine Bestätigung über die erfolgreiche Übertragung angefordert werden. Dabei muss der Projektname, die Revisionsnummer und der Safe-CPU-Name angezeigt werden. Nach dem Aufruf von [SetUserPromptTime\(\)](#) muss der Aufruf dieser Funktion innerhalb von 60 Sekunden stattfinden.

```
FUNCTION __CDECL GLOBAL P_Safety_SetConfigured
VAR_INPUT
    prjName    : ^CHAR;
    revNbr    : ^CHAR;
    scpuName  : ^CHAR;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;

#define SAFETY_SET_CONFIGURED(pCil,p1,p2,p3)
    pCil^.SetConfigured
    $ P_Safety_SetConfigured(p1,p2,p3)
```

Übergabeparameter	Typ	Beschreibung	
prjName	^CHAR	Name des Projekts, der in der Download-Datei gespeichert ist	
revNbr	^CHAR	Revisionsnummer des Projekts, die in der Download-Datei gespeichert ist	
scpuName	^CHAR	Name der Safe-CPU, der in der Download-Datei gespeichert ist	
Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	Funktion erfolgreich
		#0	Fehler-Code

Anforderungen

Connection-State == CS_LOGGED_IN und File-State == FS_DOWNLOAD.

6.4.2.22 SAFETY_SET_FILE

Gibt die Datei an, die vom Safety Designer für den Download erstellt wurde.

Setzt den File-State auf FS_FILE, wenn die Datei geöffnet und gelesen werden kann.

```
FUNCTION __CDECL GLOBAL P_Safety_SetFile
VAR_INPUT
    filename : ^CHAR;
END_VAR
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_SET_FILE(pCil,p1)
    pCil^.SetFile
    $ P_Safety_SetFile(p1)
```

Übergabeparameter	Typ	Beschreibung
filename	^CHAR	Name der Datei (0-terminierter String)
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich #0 Fehler-Code

Anforderungen

File-State == FS_IDLE

6.4.2.23 SAFETY_SET_IMAGE_FW

Gibt die Image-Datei für das Firmware-Update an.

Setzt den File-State auf FW_FILE, wenn die Datei geöffnet und gelesen werden kann.

```
FUNCTION __CDECL GLOBAL P_Set_Image_Fw
VAR_INPUT
    pImagePath : ^CHAR;
END_VAR
VAR_OUTPUT
    retval : DINT;
```

```

END_VAR;

#define SAFETY_SET_IMAGE_FW(pCil,p1)
  pCil^.SetImageFw
  $ P_Set_Image_Fw(p1)

```

Übergabeparameter		Typ	Beschreibung	
pImagePath		^CHAR	Pfad zum Image auf der Steuerung (0-terminierter String)	
Rückgabeparameter		Typ	Beschreibung	
retval		DINT	0	Funktion erfolgreich
			#0	Fehler-Code

Anforderungen

File-State == FS_IDLE

6.4.2.24 SAFETY_SET_SAFETY_NBR

Übernimmt die vom Anwender eingegebene Sicherheitsnummer der Ziel-Safe-CPU. Beim Verbindungsauflauf zur Safe-CPU wird geprüft, ob diese Sicherheitsnummer mit der Sicherheitsnummer in der Safe-CPU übereinstimmt.

Vor dem Aufruf dieser Funktion muss [SetUserPromptTime\(\)](#) aufgerufen und die Sicherheitsnummer vom Benutzer angefordert werden. Nach dem Aufruf von [SetUserPromptTime\(\)](#) muss der Aufruf dieser Funktion innerhalb von 60 Sekunden stattfinden.

Setzt Connection-State auf SAFETY_NBR

```

FUNCTION __cdecl GLOBAL P_Safety_SetSafetyNbr
VAR_INPUT
  safetyNbr : UDINT;
END_VAR
VAR_OUTPUT
  retval    : DINT;
END_VAR;

#define SAFETY_SET_SAFETY_NBR(pCil,p1)
  pCil^.SetSafetyNbr
  $ P_SetSafetyNbr(p1)

```

Übergabeparameter		Typ	Beschreibung	
safetyNbr		UDINT	Safetynummer	

Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 Funktion erfolgreich
		#0 Fehler-Code

Anforderungen

Connection-State == IDLE.

6.4.2.25 SAFETY_SET_SERVICE_MODE

Weist die Safe-CPU an, in den Run-Status SERVICE zu wechseln.

```
FUNCTION __cdecl GLOBAL P_Safety_SetServiceMode
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_SET_SERVICE_MODE(pCil)
    pCil^.SetServiceMode
    $ P_Safety_SetServiceMode()
```

Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 OK
		#0 Fehler-Code

Anforderungen

Connection-State == CS_LOGGED_IN.

6.4.2.26 SAFETY_SET_TEMP_SERVICE_MODE

Weist die Safe-CPU an, vorübergehend in den Run-Status SERVICE zu wechseln. Die Applikation wird dadurch neu gestartet.

```
FUNCTION __cdecl GLOBAL P_Safety_SetTempServiceMode
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_SET_TEMP_SERVICE_MODE(pCil)
    pCil^.SetTempServiceMode
    $ P_Safety_SetTempServiceMode()
```

Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	OK
		#0	Fehler-Code

Anforderungen

Connection-State \geq CS_CONNECTION.

6.4.2.27 SAFETY_SET_USERPROMPT_TIME

Muss aufgerufen werden, wenn vom Benutzer eine Eingabe angefordert wird. Der Zeitpunkt dieses Aufrufs wird gemerkt und später für eine Überprüfung der Plausibilität der Eingabe verwendet (es darf z.B. [SetSafetyNbr\(\)](#) nicht unmittelbar nach [SetUserPromptTime\(\)](#) aufgerufen werden). Ob es zu einem Fehler im zeitlichen Verhalten gekommen ist, kann am Rückgabewert der nachfolgend aufgerufenen Funktion abgelesen werden.

```
FUNCTION __CDECL GLOBAL P_Safety_SetUserPromptTime
VAR_OUTPUT
    retval : DINT;
END_VAR;

#define SAFETY_SET_USERPROMPT_TIME(pCil)
    pCil^.SetUserPromptTime
    $ P_Safety_SetUserPromptTime()
```

Rückgabeparameter	Typ	Beschreibung	
retval	DINT	0	OK
		#0	Fehler-Code

6.4.2.28 SAFETY_SET_VERIFIED

Setzt den Konfigurations-Status in der Safe-CPU auf „verifiziert“.

Vor dem Aufruf dieser Funktion muss [SetUserPromptTime\(\)](#) aufgerufen und vom Benutzer eine Bestätigung über den Verifizierungsvorgang angefordert werden. Dabei muss der Projektname, die Revisionsnummer und der Safe-CPU-Name angezeigt werden. Nach

dem Aufruf von [SetUserPromptTime\(\)](#) muss der Aufruf dieser Funktion innerhalb von 60 Sekunden stattfinden.

```
FUNCTION __CDECL GLOBAL P_Safety_SetVerified
VAR_INPUT
    prjName    : ^CHAR;
    revNbr    : ^CHAR;
    scpuName  : ^CHAR;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;

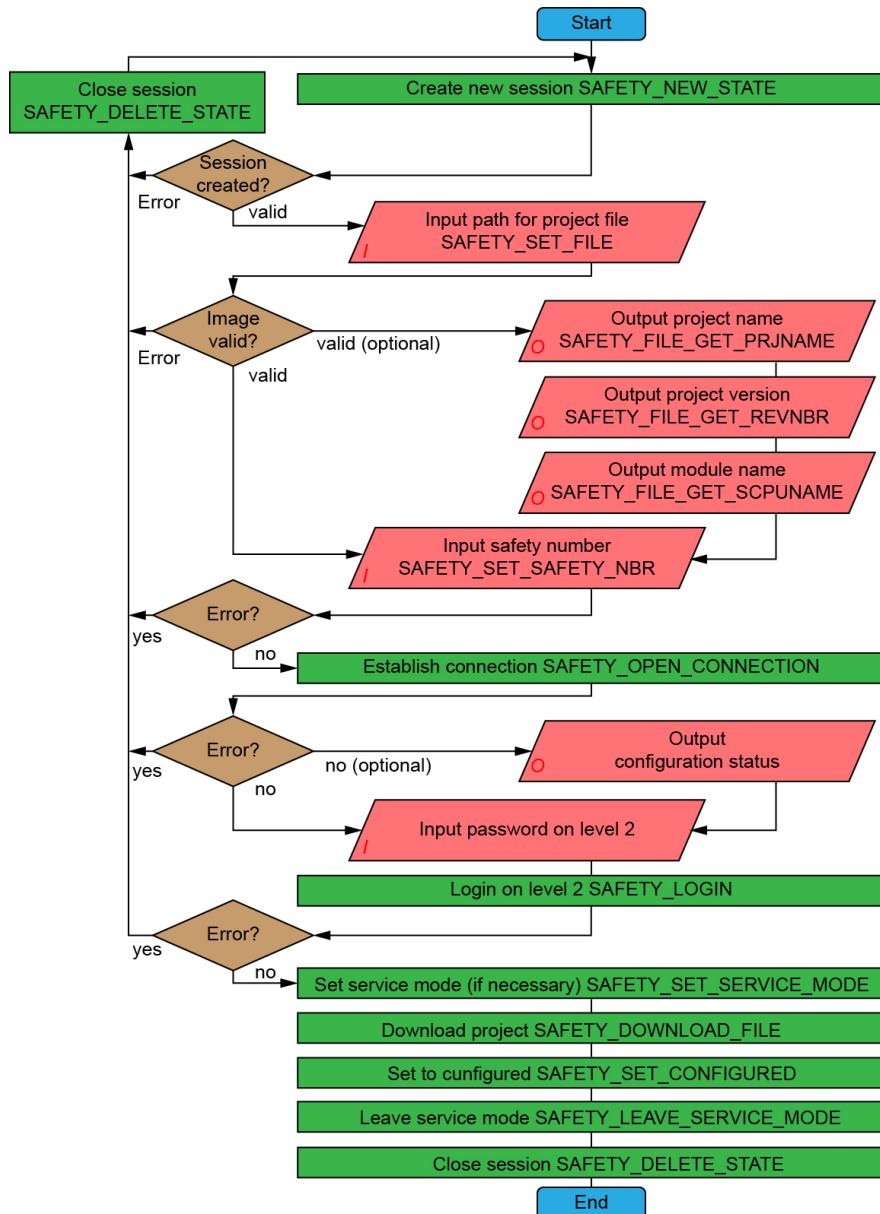
#define SAFETY_SET_VERIFIED(pCil,p1,p2,p3)
    pCil^.SetVerified
    $ P_Safety_SetVerified(p1,p2,p3)
```

Übergabeparameter	Typ	Beschreibung
prjName	^CHAR	der Name des Projekts, der in der Download-Datei gespeichert ist.
revNbr	^CHAR	die Revisionsnummer des Projekts, die in der Download-Datei gespeichert ist.
scpuName	^CHAR	der Name der Safe-CPU, der in der Download-Datei gespeichert ist.
Rückgabeparameter	Typ	Beschreibung
retval	DINT	0 OK ≠0 Fehler-Code

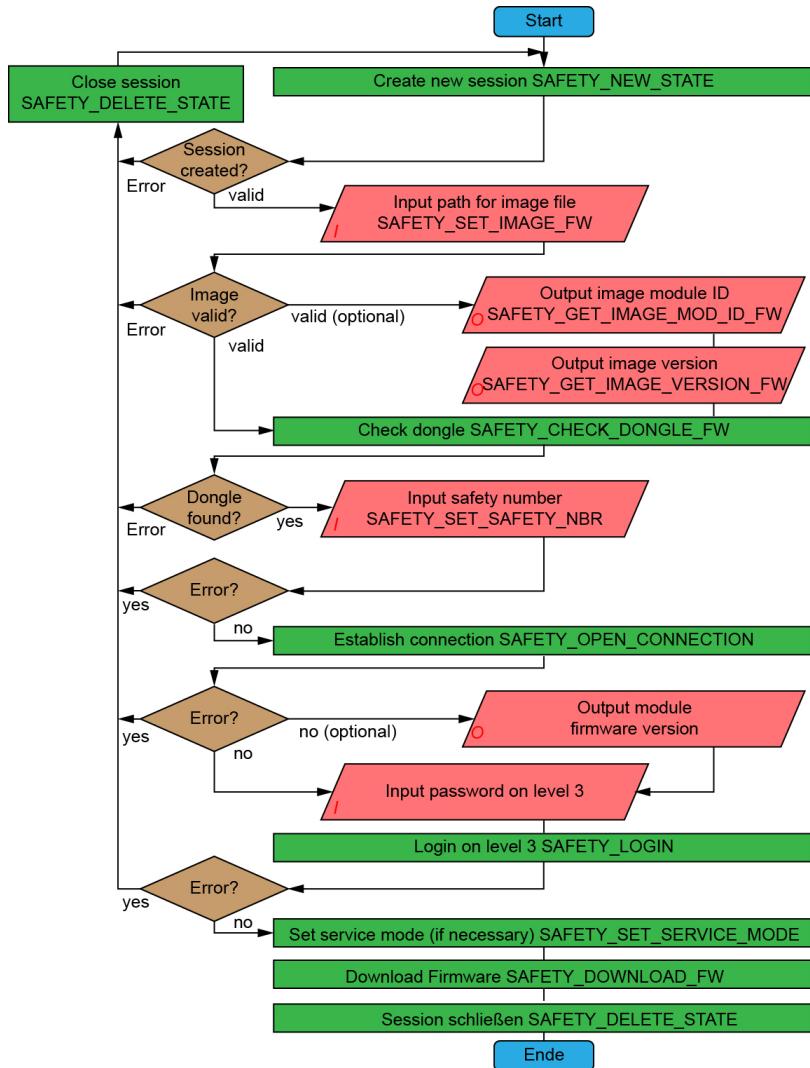
Anforderungen

Connection-State == CS_LOGGED_IN und File-State == FS_FILE.

6.4.3 Ablauf Projekt-Download



6.4.4 Ablauf Firmware-Download



6.4.5 Fehler-Codes

SAFETY_E_IN_USE	Es konnte kein Safety-State angelegt werden (mit SAFETY_NEW_STATE), weil bereits einer existiert.
SAFETY_E_OPEN	Beim Öffnen des für den Download vorgesehenen Files ist ein Fehler aufgetreten.
SAFETY_E_IO	Beim Lesen des für den Download vorgesehenen Files ist ein Fehler aufgetreten.
SAFETY_E_FILEDATA	Beim Lesen des für den Download vorgesehenen Files wurden ungültige Daten erkannt.
SAFETY_E_INVALID_USERPR OMPT_TIME	Ein Funktionsaufruf, der innerhalb einer gewissen Zeitspanne seit dem Aufruf von SAFETY_SET_USERPROMPT_TIME stattfinden muss, ist außerhalb der Zeitspanne durchgeführt worden.
SAFETY_E_INVALID_CIL_VERS ION	Die Safety-DLL konnte nicht geladen werden, weil die erforderliche Schnittstelle im LASAL-Betriebssystem nicht existiert oder eine falsche Version aufweist.
SAFETY_E_BUFFER_UNDERFL OW	Bei einem internen Funktionsaufruf wurden zu wenig Daten zurückgegeben.
SAFETY_E_BUF_TOO_SMALL	Ein als Funktionsparameter übergebener Buffer ist zu klein.
SAFETY_E_OUT_OF_MEM	Es steht zu wenig Speicher zur Verfügung (Heap).
SAFETY_E_INVALID_SAFETY_ NBR	Beim Verbindungsaufbau wurde festgestellt, dass die vom Benutzer eingegebene Sicherheitsnummer nicht mit der Sicherheitsnummer der Ziel-Safe-CPU übereinstimmt.
SAFETY_E_HWT_CMD	Beim internen Aufruf einer Hardwaretree-Funktion ist ein Fehler aufgetreten.
SAFETY_E_INVALID_PARAM	Es wurde ein ungültiger Wert eines Funktionsparameters angegeben.
SAFETY_E_MODULE_NOT_FO UND	Beim Versuch, die Verbindung aufzubauen, wurde das Ziel-Modul nicht gefunden.
SAFETY_E_INVALID_SN_RSP	Bei der Antwort, die von der Safe-CPU empfangen wurde, wurde eine ungültige Sequenznummer erkannt.
SAFETY_E_INVALID_LEN_RSP	Bei der Antwort, die von der Safe-CPU empfangen wurde, wurde eine ungültige Länge erkannt.
SAFETY_E_TOO_LESS_DATA_ RSP	In der Antwort, die von der Safe-CPU empfangen wurde, sind zu wenig Daten vorhanden.
SAFETY_E_INVALID_CRC_RSP	Bei der Antwort, die von der Safe-CPU empfangen wurde, wurde eine ungültige CRC erkannt.
SAFETY_E_INVALID_FRAME TYPE_RSP	Bei der Antwort, die von der Safe-CPU empfangen wurde, wurde ein ungültiger Frametyp erkannt.

SAFETY_E_INVALID_ADDR_RSP	Bei der Antwort, die von der Safe-CPU empfangen wurde, wurde eine ungültige Adresse erkannt.
SAFETY_E_TIMEOUT_RSP	Die Antwort, die von der Safe-CPU erwartet wird, wurde innerhalb einer gewissen Zeit nicht empfangen.
SAFETY_E_INVALID_SESSID_RSP	Bei der Antwort, die von der Safe-CPU empfangen wurde, wurde eine ungültige Session-ID erkannt.
SAFETY_E_INVALID_STATE	Ein Funktionsaufruf wurde in einem ungültigen Zustand durchgeführt. Die Funktion SAFETY_FILE_GET_PRJNAME muss beispielsweise nach der Funktion SAFETY_SET_FILE aufgerufen werden.
SAFETY_E_INVALID_STATE_1	
SAFETY_E_INVALID_STATE_2	
SAFETY_E_INVALID_STATE_3	
SAFETY_E_INVALID_STATE_4	
SAFETY_E_SSDO_RESULT	In der Antwort, die von der Safe-CPU empfangen wurde, wird im Feld Returncode ein Fehler angezeigt. Die Fehlernummer ist der Offset zur Basis 10000. Z.B.: 10123 bedeutet Fehler 123 wurde von der Safe-CPU zurückgeschickt.

7 **FTP-Server**

7.1 **Befehle**

Mit dem Lasal OS F(ile) T (ransfer) P (rotocol) können Dateien zwischen einem FTP-Client und dem FTP-Protokoll-basierenden Server ausgetauscht werden.

7.2 **Schnellstart**

Nachfolgend ist eine einfache Beschreibung zur Verwendung des FTP-Servers angeführt.

7.2.1 **Anforderungen**

LasalOS	ab Version 01.01.98 , ftpsvr.dlm Version 01.01.001 benötigt
Plattform	alle Plattformen mit einer Ethernet Schnittstelle

7.2.2 **Installation**

- Die FTPSVR.DLM muss in das C:\LSLSYS Verzeichnis kopiert werden.
- Das Standardverzeichnis des FTP-Servers ist C:\Um das Verzeichnis für alle Klienten zu ändern, muss das Standardverzeichnis des Servers konfiguriert werden.
Dies wird durch den Command Line Interface Befehl FTPSVR OPTION ermöglicht.
z.B.: FTPSVR OPTION DEFROOTDIR C:\FTP-ROOT\
Das Standardverzeichnis benötigt einen abschließenden Backslash.
Es gibt auch die Möglichkeit für jeden Client ein eigenes Standardverzeichnis anzulegen.
- Der FTP-Server kann mit dem CLI-Befehl FTPSVR START gestartet werden.
- Um einen FTP-Benutzer einzufügen, kann der CLI-Befehl FTPSVR USER ADD verwendet werden.

7.3 **Sicherheit**

7.3.1 Benutzernamen

FTP Klienten benötigen einen Benutzernamen, um sich bei einem FTP-Server einzuloggen (bei anonymen FTPs wird der Benutzername deaktiviert).

Der LasalOS FTP-Server vergleicht die Namen, wobei Groß- und Kleinschreibung nicht berücksichtigt wird.

Z.B.: Benutzername FTPUSERX ist gleich ftpuserx oder FTPUserX

7.3.2 Passwörter

Ein Passwort ist nicht zwingend notwendig.

Ein Passwort sollte immer bei FTP-Benutzern verwendet werden, die vollen Schreib- bzw. Lesezugriff auf mehrere Dateien haben.

Ein Passwort sollte auch benutzt werden, wenn der Server nur von bestimmten Benutzern verwendet werden soll.

7.3.3 Zugriffsberechtigung



Für jeden FTP-Benutzer muss eine allgemeine Zugriffsberechtigung konfiguriert werden.

Eine Allgemeine Zugriffsberechtigung erlaubt den Zugriff auf alle Laufwerke, Verzeichnisse und Dateien. Neben dem allgemeinen Zugriff können auch erweiterte Zugriffsberechtigungen konfiguriert werden.

Mittels der erweiterten Zugriffsberechtigung kann auf bestimmte Laufwerke, Verzeichnisse oder Dateien zugegriffen werden,

z.B.: ein Benutzer mit allgemeiner Zugriffsberechtigung 'R' (read-only), erweiterter Zugriffsberechtigung 'W' (write) für den Pfad C:\TEMP und erweiterter Zugriffsberechtigung 'N' für die Datei C:\TEMP\text.txt.

Im Beispiel hat der Benutzer nur Lesezugriff auf alle Laufwerke, Verzeichnisse und Dateien außer das Verzeichnis C:\TEMP und der Datei C:\TEMP\text.txt.

Der Benutzer hat vollen Zugriff (schreiben/lesen) auf den Pfad „C:\TEMP“ sowie alle Unterverzeichnisse und Dateien, aber keinen Zugriff auf die Datei C:\TEMP\text.txt.



Die erweiterten Zugriffsberechtigungen müssen in der richtigen Reihenfolge hinzugefügt werden!

D.h., der LasalOS FTP-Server durchsucht die Liste der Zugriffsberechtigungen beginnend vom ersten bis zum letzten Eintrag.

z.B.: Wenn C:\TEMP mit Zugriff „W“ der erste Eintrag ist und C:\TEMP\text.txt der zweite, dann hat der Benutzer vollen Zugriff auf C:\TEMP\ und der zweite Eintrag wird ignoriert.

Möchte ein eingeloggter FTP-Benutzer die Datei C:\TEMP\text.txt überschreiben, durchsucht der LasalOS FTP-Server die Liste der erweiterten Zugriffsberechtigungen und findet den Eintrag C:\TEMP mit Zugriffsrecht „W“.

Reihenfolge der erweiterten Zugriffsberechtigungen

C:\TEMP
C:\TEMP\text.txt

Der Eintrag „C:\TEMP\text.txt“ wird ignoriert.

Richtige Reihenfolge:

C:\TEMP\text.txt
C:\TEMP

7.3.4 Anonymer FTP

Der LasalOS FTP-Server unterstützt anonyme FTPs.

Anonyme FTPs ist die Standardkonfiguration des Servers.

Wenn die Option anonymer FTP aktiviert ist, brauchen Klienten keinen Benutzernamen und kein Passwort, um sich am FTP-Server einzuloggen.

Wenn die Option anonymer FTP aktiviert ist, wird der erweiterte FTP-Zugriff verfügbar, um die Zugriffsberechtigung für anonyme FTP-Klienten einzustellen.



Wenn die anonyme FTP-Zugriffsberechtigung auf Schreibrecht („W“) eingestellt ist, haben alle anonymen Klienten vollen Zugriff auf alle Laufwerke, Dateien und Verzeichnisse.

7.4 Server-Konfiguration

Es gibt mehrere Optionen zur Änderung der Konfiguration des LASAL OS FTP-Servers.

7.4.1 ANONYMOUS

Option zur Aktivierung oder Deaktivierung des anonymen FTPs.

Werte

0	OFF: Deaktiviert anonymen FTP (Standard)
1	ON: Aktiviert anonymen FTP

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.2 ANONYMACCESS

Option zur Einstellung der Zugangsberechtigung für anonyme FTP-Klienten.

Werte

N	FTP_ACCESS_NONE: kein Zugang
R	FTP_ACCESS_READ: nur Lesezugriff (Standard)
W	FTP_ACCESS_WRITE: Voller Zugang (lesen/schreiben/löschen)

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.3 NOFTPSINI

Option, um die Schreibberechtigung der FTP-Benutzer in die Datei ftps.ini ein- oder auszuschalten. Wird diese Option eingeschaltet (ON), gehen alle FTP-Benutzer beim Abschalten des Servers (Stopp) oder Neustart des Systems verloren.

Werte

0	OFF: ftps.ini wird verwendet (Standard)
1	ON: ftps.ini wird nicht verwendet

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.4 WINCOMPLISTFORMAT

Mit dieser Option wird es möglich das Format und den Inhalt der geschickten Daten über einen Listbefehl zu ändern.

Werte

0	OFF: Standard Listenformat
1	ON: Microsoft Windows-kompatible Listformat

Diese Option kann aktiviert werden, um die Zeitinformationen der Dateien und Ordner bei der Verwendung vom Microsoft Internet Explorer als FTP oder eines kompatiblen FTP Clients zu zeigen.

Nicht jeder FTP Client muss das Microsoft Windows kompatible Listenformat verarbeiten können, da es sich von der FTP Standardformat-Spezifikation unterscheidet.

Nach der FTP-Standard-Format-Spezifikation werden keine Zeitinformationen an den FTP-Client geschickt, wenn die aktuellen Informationen ein Jahr enthalten. Das Jahr oder die Zeitinformationen können jedoch geschickt werden, anstatt des Jahres oder einer Kombination der beiden. Der Lasal OS FTP-Server schickt standardmäßig Datumsinformationen mit der Jahresangabe.

Unterstützte FTP-Befehle

FTP-Befehl	DLL-Version	Kurzbeschreibung
QUIT	01.01.001	Logout vom Server, Benutzer beenden
HILFE	01.01.001	Informationen mit verfügbaren Befehlen an den Client schicken
NOOP	01.01.001	Keine Operation
USER	01.01.001	Benutzername, um am Server einzuloggen.
PASS	01.01.001	Benutzerpasswort
MODE	01.01.001	Übergabemodus (S-Stream, Block B-, C-Compressed), der FTP-Server unterstützt nur S
TYP	01.01.001	Darstellungstyp (A, I)0
STRU	01.01.001	Bestimmt die Dateistruktur (F-File, R-Record Struktur, P-Page-Struktur), der FTP-Server unterstützt nur F
PORT	01.01.001	Spezifikation für den Daten-Port, der zur Datenverbindung verwendet wird.
PASV	01.01.001	Abfrage eines anderen Ports als der Standardport
SYST	01.01.001	Ermittelt den Typ des Betriebssystems
DELE	01.01.001	Datei löschen
RMD	01.01.001	Verzeichnis entfernen
CDUP	01.01.001	Hauptverzeichnis ändern
CWD	01.01.001	Aktuelles Verzeichnis ändern
MKD	01.01.001	Verzeichnis erstellen
PWD	01.01.001	Gibt den Namen des aktuellen Verzeichnisses zurück
RNFR	01.01.001	Umbenennen von
RNTO	01.01.001	Umbenennen nach
SIZE	01.01.001	Dateigröße
LIST	01.01.001	Verzeichnisliste
NLST	01.01.001	Liste der Verzeichnisnamen
STOR	01.01.001	Daten empfangen und als eine Datei speichern
RETR	01.01.001	Eine Kopie einer Datei senden
APPE	01.01.001	Daten empfangen und als/in eine Datei speichern/anhängen

Anforderungen

LasalOS: ab 01.01.098, um die Option über die Methode [SetConfig](#) der _FTPServer Klasse (siehe Kapitel [LasalOS FTP-Server API](#)) zu ermöglichen oder 01.02.004, um die Option über den cli Befehl [FTPSVR OPTION](#) (siehe Kapitel [Command-Line-Interface \(CLI\) Kommandos](#)) zu ermöglichen, ab

FTP-Server-DLL-Version 01.01.004

7.4.5 PASVPORT

Mit dieser Option kann der Wert der Ports, der vom FTP-Server verwendet werden soll, eingestellt werden.

Werte

0-65535 (Standard: 0, dynamisch)

Wird der Wert 0 benutzt, dann weist der FTP-Server einen Port dynamisch zu.

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.6 DEFROOTDIR

Option zur Einstellung des Standard-Hauptverzeichnisses, das vom FTP-Server benutzt wird, wenn ein Client sein eigenes Hauptverzeichnis nicht verwendet.

Werte

ein 0-terminierter String (Standard: C:\)

Das Standardverzeichnis benötigt einen vorgestellten Backslash.



Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.7 MAXSESSIONS

Option zur Einstellung der maximalen Anzahl von Nutzern die der FTP-Server gleichzeitig erlaubt.

Werte

0 (Standard: 0, kein Grenzwert, DLL Version 01.01.001)

(Standard: 10, DLL Version > 01.01.001)

Mit einem Wert von 0 wird keine Grenze eingestellt

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.8 FORWARDSLASH

Wenn diese Option aktiviert ist, konvertiert der FTP-Server alle Backslashes in Schrägstriche bei Zurückgabe eines Pfades für einen Client.

Werte

0 OFF: nicht konvertieren (standard)

1 ON: 1Konvertiert Backslashes in Schrägstriche

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.9 BACKSLASH

Wird diese Option aktiviert, dann konvertiert der FTP-Server alle Schrägstriche in Backslashes im Dateinamen, der von einem Client empfangen wurde.

Werte

0	OFF: nicht konvertieren
1	ON: Konvertiert Schrägstriche in Backslashes (Standard)

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.10 NODRIVE

Wird diese Option aktiviert, dann entfernt der FTP-Server das Laufwerk vom Pfad mit einem PWD-Befehl.

Werte

0	OFF: nicht entfernen
1	ON: entfernt Laufwerkbsbuchstabe (Standard)

Wenn der Laufwerksbuchstabe und der Doppelpunkt nicht entfernt wurden, dann ist der Server mit allen Unix-basierten Clients nicht kompatibel.

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.11 CMDTMO

Option zur Änderung des Befehle-Timeouts.

Wartezeit für einen Clientbefehl.

Werte

0-60 **FTPSVR_INF**(Standard: 30)

Einheit

Sekunden

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.12 WRITETMO

Option zur Änderung des Schreib-Timeouts.

Wartezeit für Daten an einen abgesetzten Host schicken.

Werte

0-60 **FTPSVR_INF** (Standard: 30)

Einheit

Sekunden

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.13 READTMO

Option zur Änderung des Lese-Timeouts.

Wartezeit zum Empfangen von Daten (Dateien) und Quittierung von einem abgesetzten Host.

Werte

0-60 **FTPSVR_INF** (default: 30)

Einheit

Sekunden

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.14 TPrio

Option zur Änderung der Priorität des Verbindungstasks.

Werte

1-14 (Standard: 8)



Diese Option sollte sehr sorgfältig verwendet werden.

Die Priorität der Verbindungstasks darf gleich hoch sein wie die Priorität des FTP-Servers Daemon aber nicht höher.

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.4.15 DPrio

Option zur Änderung der Priorität der FTP-Servers Daemon Tasks (Hauptaufgabe des Servers zum Abfragen der Eingangsverbindungen)

Werte

1-14 (Standard: 9)



Diese Option sollte nur mit Vorsicht verwendet werden!

Der Background Task hat eine höhere Priorität als die Standardeinstellung des FTP-Servers Daemon. Falls die Restzeit für den FTP-Server Daemon

wegen des Background-Tasks unzureichend ist, sollte die Priorität des Daemon Tasks auf die gleiche des Background Tasks erhöht werden.

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.5 Command Line Interface (CLI) Befehle

7.5.1 FTPSVR INFO

Informationen über den FTP-Server, den Status, registrierter Benutzer, aktive Verbindungen, die DLL-Version und der aktuellen FTP-Server Konfiguration.

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.5.2 FTPSVR HELP

Informationen über alle verfügbaren FTP-Server CLI-Befehle und ihrer Verwendung.

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.5.3 FTPSVR START

Startet den FTP-Server. Der Befehl kann über das Command Line Interface oder autoexc.lsl direkt ausgeführt werden.

Ist der FTP-Server gestartet, wird die ftps.ini Datei, die alle zuvor registrierten und gespeicherten FTP-Benutzer enthält, geladen.

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.5.4 FTPSVR OPTION

Dieser Befehl wird zum Ändern oder Zeigen der Konfiguration des FTP-Servers verwendet.

Wenn kein Parameter angegeben wurde, zeigt dieser die aktuelle Konfiguration.

Optionale Parameter

<option>	Option zum Konfigurieren
<value>	Der Wert der spezifizierten Option
<help>	Listet alle verfügbaren Optionen und ihre möglichen Werte auf.

Für verfügbare Optionen und ihre möglichen Werte „siehe Kapitel Server Konfiguration“.

Ist der Parameter Option ohne einen Wert eingegeben, wird der aktuelle Wert der Option angezeigt.

Beispiel

Liste der aktuellen FTP-Server Konfiguration.

```
C:\> FTPSVR OPTION
```

```
Anonymous FTP .....: OFF
No ftps.ini (user config file) .....: OFF
FTP-Server daemon priority .....: 9
FTP-Server task priority (connection) .....: 8
Read timeout .....: 30
Write timeout .....: 30
Command timeout .....: infinite
No drive on PWD command .....: ON
Convert "/" to "\\" (receiving) .....: ON
Convert "\\" to "/" (sending) .....: OFF
Port value for passive data connections ...: dynamic
Max. number of FTP sessions .....: no limit
FTP-Server default root directory .....: C:\
FTP user access right added successfully
```

```
C:\>
```

Aktuelle Konfiguration der anonymen FTP-Option.

```
C:\> FTPSVR OPTION ANONYMOUS
Anonymous FTP .....: OFF
```

```
C:\>
```

Aktiviert den anonymen FTP.

```
C:\> FTPSVR OPTION ANONYMOUS ON
```

C:\>

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001.

7.5.5 FTPSVR ACCESS

Dieser Befehl wird zum Einfügen und Entfernen der Zugangsberechtigungen für FTP-Clients verwendet.

Optionale und benötigte Argumente.

ADD <username>

Fügt eine Zugriffsberechtigung für einen bestimmten FTP-Benutzer <username> ein und speichert diese in der ftps.ini Datei.

<username>	Der FTP-Benutzer für den die Zugangsberechtigung hinzugefügt wird
------------	-------------------------------------------------------------------

Beispiel

```
C:\> FTPSVR ACCESS ADD FTPuser1
```

```
Expanded access rights: No path to exit, no position to append entry
```

```
Access path.....: C:\LSLSYS
Access right (N,R,W)...: R
Insert at position....:

FTP user access right added successfully

Access path.....:

Saving FTP user... successful
```

```
C:\>
```

Access path	Der angegebene Pfad für den das folgende Zugriffsrecht gültig ist
Zugriffsrecht	Die Zugangsberechtigung für den oben angeführten Pfad
Insert at position	Die Position, an dem der Eintrag eingefügt werden soll (keine Position, um den Eintrag anzuhängen)

Siehe Kapitel SICHERHEIT für weitere Informationen über Zugangsberechtigungen.

Der neu eingefügte Eintrag wird in die Datei `ftps.ini` gespeichert, wenn die Option `FTPServer NOFTPSINI` auf `OFF` eingestellt ist (Standard). Die gespeicherten FTP-Benutzer und ihrer Zugangsberechtigungen werden mit dem Neustart des Servers geladen.

REMOVE <username> <path>

Entfernt eine Zugangsberechtigung für einen bestimmten FTP-Benutzer, der im Pfad angegeben wurde.

<username>	Der FTP-Benutzer für den die Zugangsberechtigung entfernt werden soll
------------	-----------------------------------------------------------------------

| <path> | Der Pfad der Zugangsberechtigung die entfernt werden soll. |

Beispiel

```
C:\> FTPSVR ACCESS REMOVE FTPuser1 C:\LSLSYS
```

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001.

7.5.6 FTPSVR USER

Dieser Befehl fügt FTP-Clients hinzu, entfernt und liefert Informationen über FTP-Clients.

Optionale und benötigte Argumente:

INFO <username> </p>

Zeigt Informationen des FTP Benutzers.

Wenn kein Parameter eingetragen ist, zeigt der Befehl die Informationen aller registrierten FTP-Clients.

Optionale Parameter

<username>	zeigt die Informationen des FTP Benutzers <username>
------------	------------------------------------------------------

| </p> | stoppt Ausgabe nach jeder Vollbildseite |

Beispiel

```
C:\> FTPSVR USER INFO
```

```
Username ..... FTPuser1
General access right ..... Read-only [R]
```

```
Root directory ..... Not set, using FTP-Server default
Reference ..... Operating system
Status ..... STORED
LoggedIn ..... 0
Enlarged access rights ..... Not set

Username ..... FTPUser2
General access right ..... No access [N]
Root directory ..... C:\TEMP\
Reference ..... Operating system
Status ..... STORED
LoggedIn ..... 0
Expanded access rights

[W] ... C:\TEMP\
```

C:\>

Der Befehl kann nicht aus der autoexec.lsl ausgeführt werden.

ADD

Dieser Befehl fügt einen FTP-Benutzer ein und hat keine Parameter. Der Benutzer wird aufgefordert die benötigten Informationen einzugeben, um einen FTP-Benutzer nach dem Ausführen des Befehls anzulegen. Der Befehl kann nicht aus der autoexec.lsl ausgeführt werden.

Beispiel

```
C:\> FTPSVR USER ADD

Username (max. 63)....: FTPUser1
Password (max. 63)....: test123
General Access (N,R,W): W
Root directory....:

FTP user created successfully

Expanded access rights: No path to exit, no position to append entry

Access path.....:

Saving FTP user... successful

C:\>
```

benutzername Der Benutzername, den der FTP-Client verwenden kann, um eine Verbindung mit dem Server aufzubauen (die folgenden Zeichen sind nicht erlaubt '/', '\')

Passwort	Das Passwort, der zum Login als bestimmter Benutzer verwendet werden muss. Ein Passwort ist nicht zwingend notwendig. Soll sich ein FTP-Benutzer ohne Passwort einloggen können, muss kein Passwort eingetragen werden.
General Access	Der allgemeine Zugang zum FTP-Client (diese Art des Zugriffs ist gültig für alle Laufwerke, Verzeichnisse und Dateien).
Root directory	Das Hauptverzeichnis, das für den FTP-Client verwendet wird. Ist kein Hauptverzeichnis angegeben, wird das Hauptverzeichnis des Servers verwendet.

Für erweiterten Zugangsberechtigungen, siehe **FTPSVR ACCESS**

Der neu hinzugefügte Eintrag wird in der Datei **ftps.ini** gespeichert, wenn die Option **FTP-Server NOFTPSINI** auf **Off** eingestellt ist (Standard). Die gespeicherten FTP-Benutzer werden beim Neustart des Servers geladen.

REMOVE <username>

Entfernt einen FTP-Benutzer und löscht ihn aus der **ftps.ini** Datei.

<username> Der zu entfernende FTP-Benutzer (Parameter benötigt)

Beispiel

```
C:\> FTPSVR USER REMOVE FTPuser1
```

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001.

7.5.7 FTSPVR STOP

Stoppt den FTP-Server und trennt alle aktiven FTP-Clients.

Anforderungen

LasalOS: ab 01.01.098, FTP-Server DLL ab Version 01.01.001

7.6 Lasal OS FTP-Server API

7.6.1 OS Interface Library Class _FTPServer

Die Schnittstelle des Betriebssystems wird von der OS-Interface Library durch die _FTPServer Klasse zur Verfügung gestellt. Die OS-Interface Library ist für LASAL CLASS 1 und LASAL CLASS 2 verfügbar.

Um die Schnittstelle zu benutzen, muss ein neues Projekt erstellt und die OS-Klasse _FTPServer Interface Library importiert werden. Eine neue Klasse mit einem Objekt der _FTPServer Klasse muss erstellt werden.

Die Methoden der Schnittstelle können durch den erstellten Objektkanal aufgerufen werden.

Beispiel

Objektkanal der neu erstellten Klasse: ocFTPSvr

Methode aufrufen: ocFTPSvr.methodname(Parameter, ...)

7.6.1.1 AddAccess

Fügt eine erweiterte Zugangsberechtigung zu einem bestehenden FTP-Benutzer hinzu.

```
FUNCTION __CDECL VIRTUAL GLOBAL AddAccess
VAR_INPUT
    Username    : ^CHAR;
    Access      : UDINT;
    Path        : ^CHAR;
    InsertPos   : UDINT;
END_VAR
VAR_OUTPUT
    drc (EAX)  : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
Benutzername	^CHAR	0-terminierter String, der den Namen des FTP-Benutzers beinhaltet, zu dem eine erweiterte Zugangsberechtigung hinzugefügt werden soll.
Zugriff	UDINT	Zugriffsrecht FTP_ACCESS_NONE - no access FTP_ACCESS_READ - read-only access FTP_ACCESS_WRITE - full access (read/write/delete)
Path	^CHAR	0-terminierter String, der den Pfad oder Dateinamen enthält, für die die Zugangsberechtigung von dem zuvor genannten Parameter Access gültig ist.

		Die Zugangsberechtigung gilt für den Pfad und alle Dateien und Verzeichnisse, die der Pfad enthält. Ein abschließender Backslash wird für den Pfad nicht benötigt.				
InsertPos :	UDINT	Die Position, auf die die Zugangsberechtigung hinzugefügt werden soll. 1 = erste, 2 = zweite,...0 um den Eintrag hinzufügen				
Rückgabeparameter	Typ	Beschreibung				
dRC	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: center;">0</td><td>Erfolg</td></tr> <tr> <td style="text-align: center;"><0</td><td>Negativer Fehler-Code</td></tr> </table>	0	Erfolg	<0	Negativer Fehler-Code
0	Erfolg					
<0	Negativer Fehler-Code					

Es ist nicht möglich eine Zugangsberechtigung hinzuzufügen, wenn ein FTP-Client mit dem Namen des eingetragenen FTP-Benutzers eingeloggt ist, dessen Zugangsberechtigung erweitert werden soll.

Ist der FTP-Benutzer mittels der Methode [AddUser\(\)](#) eingefügt worden und der Parameter Save auf 1 gesetzt, so ist es möglich die Änderungen in der Datei ftps.ini durch den Aufruf der Methode [UpdateFTPSini\(\)](#) zu speichern.

Siehe Kapitel - SICHERHEIT für weitere Informationen über Zugangsberechtigungen.

Beispiel

```
dRC : DINT;

// fügen erweiterte Zugriffsrechte für den Benutzer „FTPUser“ ein

// Vollen Zugriff auf das Verzeichnis "C:\TEMP", aber nur Leserechte auf der Datei
// „C:\TEMP\test.txt“
dRC := ocFTPSvr.AddAccess("FTPUser", FTP_ACCESS_READ, "C:\TEMP\test.txt", 0);
if dRC = 0 then
// Zugriffsrechte erfolgreich eingefügt

// add next
dRC := ocFTPSvr.AddAccess("FTPUser", FTP_ACCESS_WRITE, "C:\TEMP", 0);
if dRC = 0 then
// Zugriffsrechte erfolgreich eingefügt
end_if;
end_if;
if dRC <> 0 then
// error occurred
end_if;
```

7.6.1.2 AddUser

Fügt einen FTP-Benutzer in der registrierten Liste der verfügbaren FTP-Benutzer hinzu.

```
FUNCTION __CDECL VIRTUAL GLOBAL AddUser
VAR_INPUT
    Benutzername    : ^CHAR;
    Kennwort        : ^CHAR;
    GeneralAccess  : UDINT;
    RootDir         : ^CHAR;
    Save            : USINT;
END_VAR
VAR_OUTPUT
    dRC (EAX)      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung		
Benutzername	^CHAR	0-terminierter String, der den einzufügenden Benutzernamen enthält; max. FTPSVR_USERNAME_LEN Zeichen		
Kennwort	^CHAR	0-terminierter String, der das Passwort enthält; max. FTPSVR_PASSWORD_LEN Zeichen. Ist kein Passwort gewünscht, kann ein NIL-Zeiger zugewiesen werden.		
GeneralAccess	UDINT	Die allgemeine Zugangsberechtigung der FTP-Benutzer FTP_ACCESS_NONEkein Zugang FTP_ACCESS_READNur Lesezugriff FTP_ACCESS_WRITEVoller Zugang (lesen/schreiben/löschen)		
RootDir	^CHAR	0-terminierter String, der das Hauptverzeichnis der FTP-Benutzer mit max. 260 Zeichen enthält. Wenn kein einzelnes Hauptverzeichnis erforderlich ist, kann ein NIL-Zeiger zugeordnet werden. In solchen Fällen wird das Standard-Hauptverzeichnis des Servers verwendet.		
Save	USINT	Setzt eine Flag um den Eintrag in der Datei ftps.ini durch den Aufruf der Methode UpdateFTPSInit() zu speichern. <table border="1" data-bbox="459 1095 1019 1174"> <tr> <td>0 Eintrag nicht speichern</td> </tr> <tr> <td>1 Speichert den Eintrag</td> </tr> </table>	0 Eintrag nicht speichern	1 Speichert den Eintrag
0 Eintrag nicht speichern				
1 Speichert den Eintrag				
Rückgabeparameter	Typ	Beschreibung		
dRC	DINT	0 Erfolg <0 Negativer Fehler-Code		

Wurde die Methode erfolgreich ausgeführt, kann ein FTP-Client sich mit dem Benutzernamen und Passwort des FTP-Benutzers in den FTP-Server einloggen.

Ist das Save Parameter 0, wird der FTP-Benutzereintrag beim Reset der Applikation entfernt. Ist der Parameter 1, wird der Benutzer beim Reset der Applikation nicht entfernt. Um den Eintrag zu speichern der mit dem nächsten FTP-Server Start oder Reboot verfügbar sein soll, muss die [UpdateFTPSini\(\)](#)-Methode aufgerufen werden.

Die allgemeine Zugangsberechtigung gilt für alle Laufwerke, Verzeichnisse und Dateien. Ist der Parameter GeneralAccess `FTP_ACCESS_WRITE`, hat der Benutzer uneingeschränkten Zugriff auf alle Laufwerke, Verzeichnisse und Dateien.

Es gibt die Möglichkeit, erweiterte Zugriffsrechte für die Laufwerke, Verzeichnisse und Dateien einzustellen (siehe [AddAccess\(\)](#)).

Die folgenden Zeichen sind in einem Benutzernamen nicht erlaubt: /, \

Beispiel

```
dRC      : DINT;

dRC := ocFTPSvr.AddUser("FTPUser", "123", FTP_ACCESS_READ, NIL, 0);
if dRC = 0 then
// Add user fertig
else
// Add user fehlgeschlagen
end_if;
```

7.6.1.3 EditAccess

Diese Anweisung ändert eine bestehende, erweiterte Zugangsberechtigung.

```
FUNCTION __cdecl virtual global EditAccess
VAR_INPUT
  Benutzername  : ^CHAR;
  Access        : UDINT;
  Path          : ^CHAR;
  NewPath       : ^CHAR;
  InsertPos     : UDINT;
END_VAR
VAR_OUTPUT
  dRC (EAX)    : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
Benutzername	^CHAR	0-terminierter String, der den Namen des FTP-Benutzers beinhaltet zu dem eine erweiterte Zugangsberechtigung hinzugefügt werden soll.

Zugriff	UDINT	Zugriffsrecht FTP_ACCESS_NONE kein Zugang FTP_ACCESS_READ Nur Lesezugriff FTP_ACCESS_WRITE Voller Zugang (lesen/schreiben/löschen)						
Path	^CHAR	0-terminierter String, der den Pfad oder Dateinamen enthält, für die die Zugangsberechtigung von dem Parameter Access gültig ist. Die Zugangsberechtigung gilt für den Pfad und für alle Dateien und Verzeichnisse, die den Pfad enthalten. Ein abschließender Backslash wird für die Pfade nicht benötigt.						
NewPath	^CHAR	0-terminierter String zu einem neuen Pfad. Ist dieser Parameter ein NIL-Zeiger, bleibt der Pfad unverändert.						
InsertPos :	UDINT	Die Position, auf die die Zugangsberechtigung hinzugefügt werden soll. 1 = erste, 2 = zweite,0, um den Eintrag anzuhängen						
Rückgabeparameter	Typ	Beschreibung						
dRC	DINT	<table border="1"> <tr> <td>0</td><td>Benutzer erfolgreich geändert</td></tr> <tr> <td>FTPSVR_ERR_EXISTS</td><td>Keine Änderungen an den bestehenden Benutzern</td></tr> <tr> <td><0</td><td>Negativer Fehler-Code</td></tr> </table>	0	Benutzer erfolgreich geändert	FTPSVR_ERR_EXISTS	Keine Änderungen an den bestehenden Benutzern	<0	Negativer Fehler-Code
0	Benutzer erfolgreich geändert							
FTPSVR_ERR_EXISTS	Keine Änderungen an den bestehenden Benutzern							
<0	Negativer Fehler-Code							

Es ist nicht möglich ein bestehendes Zugangsrecht zu ändern, wenn ein FTP-Client mit dem Benutzernamen des FTP-Benutzereintrags eingeloggt ist, für den die Zugriffsrechte geändert werden sollen.

Ist der FTP-Benutzer mittels der Methode [AddUser\(\)](#) eingefügt und der Parameter Save auf 1 gesetzt, ist es möglich die Änderungen in der Datei ftps.ini durch den Aufruf der Methode [UpdateFTPSini\(\)](#) zu speichern.

Nur bestehenden Zugriffsrechte können geändert werden.

Die Parameter müssen anders als die bestehenden Zugriffsrechte sein, um ein Zugriffsrecht zu ändern, sonst wird der Fehler FTPSVR_ERR_EXISTS zurückgegeben.

Wenn nur der Access Parameter sich von den zuvor eingefügten oder geänderten Zugangsrechten unterscheidet und der Pfad der gleiche ist oder ein NIL-Zeiger zugewiesen wurde, um anzugeben, dass keine Änderungen für den Pfad des Zugriffsrechts angegeben wurden, so wird der Parameter InsertPos ignoriert.

Beispiel

dRC : DINT;

```
// Bearbeiten das bestehende Zugriffsrecht (C: \ TEMP, voller Zugang)
dRC := ocFTPSvr.EditAccess("FTPUser", FTP_ACCESS_READ, "C:\TEMP", 0);
if dRC = 0 then
// Zugriffsrechte erfolgreich eingefügt
else
// Fehler aufgetreten
end_if
```

7.6.1.4 EditUser

Diese Anweisung wird verwendet um einen bestehenden, mit der Methode [AddUser\(\)](#) hinzugefügten FTP-Benutzer, zu ändern.

```
FUNCTION __cdecl virtual global EditUser
VAR_INPUT
    Benutzername      : ^CHAR;
    Benutzername      : ^CHAR;
    Kennwort          : ^CHAR;
    GeneralAccess    : UDINT;
    RootDir          : ^CHAR;
    Flags             : UDINT;
END_VAR
VAR_OUTPUT
    dRC (EAX)        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
Benutzername	^CHAR	0-terminierter String, der den zu ändernden Namen des FTP-Benutzers enthält. Die maximale Länge wird durch FTPSVR_USERNAME_LEN ermittelt.
NewUsername	^CHAR	0-terminierter String mit einem neuen Benutzernamen. Ist dieser Parameter ein NIL-Zeiger, bleibt der Benutzername unverändert.
Password	^CHAR	0-terminierter String, der das Passwort enthält. Die maximale Länge wird durch FTPSVR_PASSWORD_LEN ermittelt. Ist kein Passwort gewünscht, kann ein NIL-Zeiger zugewiesen werden.
GeneralAccess	UDINT	Die allgemeine Zugangsberechtigung der FTP-Benutzer FTP_ACCESS_NONEkein Zugang FTP_ACCESS_READNur Lesezugriff FTP_ACCESS_WRITEVoller Zugang (lesen/schreiben/löschen)
RootDir	^CHAR	0-terminierter String der das Hauptverzeichnis des FTP-Benutzers mit einem Maximum von 260 Zeichen enthält. Wenn kein einzelnes Hauptverzeichnis erforderlich ist, kann ein NIL-Zeiger zugeordnet werden. In solchen Fällen wird das Standard-Hauptverzeichnis des Servers verwendet.

Flags	UDINT	Legt die zu ändernden Parameter fest FTPSVR_CHANGE_PASSWORD - Passwort ändern FTPSVR_CHANGE_GENERALACCESS - Änderung des allgemeinen Zugangs FTPSVR_CHANGE_ROOTDIR - Hauptverzeichnis ändern								
Rückgabeparameter	Typ	Beschreibung								
dRC	DINT	<table border="1"> <tr> <td>0</td><td>Benutzer erfolgreich geändert</td></tr> <tr> <td>FTPSVR_ERR_</td><td>Keine Änderungen an den bestehenden Benutzern</td></tr> <tr> <td>EXISTS</td><td></td></tr> <tr> <td><0</td><td>Negativer Fehler-Code</td></tr> </table>	0	Benutzer erfolgreich geändert	FTPSVR_ERR_	Keine Änderungen an den bestehenden Benutzern	EXISTS		<0	Negativer Fehler-Code
0	Benutzer erfolgreich geändert									
FTPSVR_ERR_	Keine Änderungen an den bestehenden Benutzern									
EXISTS										
<0	Negativer Fehler-Code									

Nur bestehende FTP-Benutzer können geändert werden.

Die Parameter müssen von den vorhandenen FTP-Benutzern verschieden sein um einen FTP-Benutzereintrag unabhängig von dem Parameter Flag zu ändern, sonst wird Fehler FTPSVR_ERR_EXISTS zurückgeliefert.

Wurde der FTP-Benutzer mittels der Methode [AddUser\(\)](#) eingefügt und der Parameter Save auf 1 gesetzt, ist es möglich die Änderungen in der Datei ftps.ini durch den Aufruf der Methode [UpdateFTPSini\(\)](#) zu speichern.

Es ist nicht möglich einen FTP-Benutzer zu ändern, wenn ein FTP-Client mit dem Namen des FTP-Benutzereintrags, der geändert werden soll, eingeloggt ist.

Beispiel

```

dRC      : DINT;

// der Benutzername eines bestehenden Eintrags ändern
dRC := ocFTPSvr>EditUser("FTPUser", "NewFTPUser", NIL, 0, NIL, 0);

// Keine Flags angegeben, Passwort, allgemeiner Zugang, Hauptverzeichnis
// bleibt ungeändert

if dRC = 0 then
  // FTP erfolgreich geändert
else
  // Fehler aufgetreten
end_if

// Das Passwort des vorigen Benutzers könnte auch geändert werden
// Durch den Aufruf der Methode EditUser
dRC := ocFTPSvr>EditUser("NewFTPUser",
  NIL,
  "456",

```

```

0,
NIL,
FTPSVR_CHANGE_PASSWORD) ;

if dRC = 0 then
// FTP erfolgreich geändert
else
// Fehler aufgetreten
end_if;

```

7.6.1.5 GetConfig

Diese Anweisung liest eine Option aus der aktuellen FTP-Server-Konfiguration.

```

FUNCTION __CDECL VIRTUAL GLOBAL GetConfig
VAR_INPUT
    pValue      : pVoid;
    Size        : UDINT;
    Option      : UDINT;
END_VAR
VAR_OUTPUT
    dRC (EAX)   : DINT;
END_VAR;

```

Übergabeparameter	Typ	Beschreibung	
pValue	pVoid	Zeiger auf die Variable, die den Wert der angegebenen Option abruft.	
Size	UDINT	Größe der Variablen, Array oder des Speichers in Byte, auf den pValuepoints zeigt.	
Option	UDINT	Die abzurufende Option (Siehe Anhang A - Typen für eine Liste der verfügbaren Option)	
Rückgabeparameter	Typ	Beschreibung	
dRC	DINT	0 Erfolg <0 Negativer Fehler-Code	

Anschließend ist eine Liste der verfügbaren Optionen zu finden und die Größe für jede Option, um den Wert abzurufen.

Option	Typ der pValue	Size	DLL Version
FTPS_OPT_READ_TMO	DINT	4	01.01.001
FTPS_OPT_WRITE_TMO	DINT	4	01.01.001
FTPS_OPT_CMD_TMO	DINT	4	01.01.001

FTPS_OPT_NO_DRIVE	DINT	4	01.01.001
FTPS_OPT_BACKSLASH	DINT	4	01.01.001
FTPS_OPT_FORWARDSLASH	DINT	4	01.01.001
FTPS_OPT_MAX_SESSIONS	DINT	4	01.01.001
FTPS_OPT_DEF_ROOT_DIR	ARRAY OF CHAR, ^ CHAR	variable	01.01.001
FTPS_OPT_PASV_PORT	DINT	4	01.01.001
FTPS_OPT_ANONYMOUS	USINT	1	01.01.001
FTPS_OPT_ANONYMACCESS	USINT	1	01.01.001
FTPS_OPT_NO_FTPSINI	USINT	1	01.01.001
FTPS_OPT_SVR_D_PRIO	USINT	1	01.01.001
FTPS_OPT_SVR_T_PRIO	USINT	1	01.01.001

Die Größe der Option `FTPS_OPT_DEF_ROOT_DIR` ist variabel, jedoch muss es mindestens die Länge des Hauptverzeichnis plus 1 Byte für die 0-Terminierung tragen.

Das Hauptverzeichnis darf bis zu 260 Zeichen (Byte) enthalten.

Beispiel

```

dRC          : DINT;
anonymousFTP : USINT;
RootDir      : ARRAY [0..26] OF CHAR;

// ANONYM Option abfragen(anonymous FTP)

// Parameter pValue verwenden
dRC := ocFTPSvr.GetConfig(#anonymousFTP, 0,
dRC := ocFTPSvr.SetConfig("ON", 0, FTPS_OPT_ANONYMOUS);

// Standard Hauptverzeichnis abfragen
dRC := ocFTPSvr.SetConfig("C:\FTPROOT", 0, FTPS_OPT_DEF_ROOT_DIR);

```

7.6.1.6 GetUpdateFTPSiniStatus

Mit diesem Befehl wird der Status des Schreibprozesses der `ftps.ini` Datei bei Verwendung der [UpdateFTPSini\(\)](#) im Non-Blocking Modus aufgerufen.

```

FUNCTION __CDECL VIRTUAL GLOBAL GetUpdateFTPSiniStatus
VAR_INPUT
  pRC      : ^DINT;
END_VAR
VAR_OUTPUT

```

```
dRC (EAX) : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
prC	^DINT	Zeiger auf einen DINT Wert zum Abruf des Rückgabewerts der UpdateFTPSini Methode.	
Rückgabeparameter	Typ	Beschreibung	
dRC	DINT	0	Methode fertig
		FTPSVR_ERR_UPD_BUSY	Schreibprozess läuft
		FTPSVR_ERR_UPD_NO_TASK	Methode wird wieder aufgerufen, nachdem 0 zurückgegeben wurde

Die Methode sollte aufgerufen werden, solange der Rückgabewert **FTPSVR_ERR_UPD_BUSY** ist.

Der Inhalt des DINT-Werts, zu dem das prC zeigt, bleibt unverändert, bis der interne Task fertig ist und die Methode 0 zurückliefert.

Beispiel

```
dResult : DINT;
dRC      : DINT;

case m_Step of

  1:   dRC := ocFTPServer.UpdateFTPSini(1);
  m_step += 1;
  2:   dRC := ocFTPServer.GetUpdateFTPSiniStatus(#dResult);
        if dRC < 0 & dRC >> FTPSVR_ERR_UPD_BUSY then
          m_step := 0; // error occurred
        end_if;
        if dRC = 0 then
          // done, see if internal write process was successful
  if dResult < 0 then
    // error
    else
      m_step := 0; // writing FTP users successful to ftps.ini
      end_if;
    end_if;
  end_case;
```

7.6.1.7 GetUserAccessInfo

Erweiterte Zugangsrechte für einen bestimmten FTP-Benutzer einsehen.

```
FUNCTION __CDECL VIRTUAL GLOBAL GetUserAccessInfo
VAR_INPUT
    Benutzername : ^CHAR;
    Path         : ^CHAR;
    SizeOfPath   : UDINT;
    Zugriff      : ^UDINT;
    AccessIdx    : UDINT;
END_VAR
VAR_OUTPUT
    dRC (EAX)   : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung						
Benutzername	^CHAR	0-terminierter String, der den Namen des FTP-Benutzers enthält, dessen erweiterten Informationen abgerufen werden.						
Path	^CHAR	Zeiger auf einen Array oder Speicherbereich, der den Pfad des FTP-Benutzers mit erweitertem Zugriffsrecht abruft.						
SizeOfPath :	UDINT	Größe des Arrays in dem der Pfad gespeichert werden kann.						
Zugriff	^UDINT	Zeiger auf eine Variable vom Typ UDINT, der den Wert des Zugriffsrechts abruft.						
AccessIdx :	UDINT	Index der zu erhaltenen Zugangsberechtigung des FTP-Benutzers (Beginn mit 0)						
Rückgabeparameter	Typ	Beschreibung						
dRC	DINT	<table border="1"> <tr> <td>0</td><td>FTP Benutzer wurde erfolgreich gefunden</td></tr> <tr> <td>FTPSVR_ERR_NOT_FOUND</td><td>FTP-Benutzer nicht vorhanden ist oder keine Zugriffsrechte sind verfügbar</td></tr> <tr> <td>NOT_FOUND</td><td></td></tr> </table>	0	FTP Benutzer wurde erfolgreich gefunden	FTPSVR_ERR_NOT_FOUND	FTP-Benutzer nicht vorhanden ist oder keine Zugriffsrechte sind verfügbar	NOT_FOUND	
0	FTP Benutzer wurde erfolgreich gefunden							
FTPSVR_ERR_NOT_FOUND	FTP-Benutzer nicht vorhanden ist oder keine Zugriffsrechte sind verfügbar							
NOT_FOUND								

Um die erweiterten Zugangsrechte für alle FTP-Benutzer abzurufen, muss die Methode mit `FTPSVR_ERR_NOT_FOUND` aufgerufen werden.

Mit jedem Anruf muss der Parameter `AccessIdx` sich um eins erhöhen.

Ein `AccessIdx` von 0 liefert die Informationen über die ersten Zugriffsrechte einem FTP-Benutzer zurück.

Um die vollständigen Informationen aller FTP-Benutzer abzurufen, soll die Methode [GetUserInfo\(\)](#) aufgerufen werden, um die ersten FTP-Benutzer abzufragen. Wenn ein FTP-Benutzer gefunden wurde, können die Methoden [GetUserExtendedInfo\(\)](#) und

GetUserAccessInfo() aufgerufen werden, um den Rest der FTP-Benutzer-Informationen abzufragen.

Wenn die [GetUserInfo\(\)](#)-Methode aufgerufen werden kann, kann der Parameter UserIdx um eins erhöht werden, um nach dem nächsten FTP-Benutzer zu suchen.

Beispiel

```
dRC      : DINT;
Path     : ARRAY [0..260] OF CHAR;
Zugriff  : UDINT;

// die ersten erweiterten Zugriffsrechte für den FTP-Benutzer „FTPUser“ abrufen
dRC = ocFTPSvr.GetUserAccessInfo("FTPUser",
                                  #Path[0],
                                  sizeof(Path),
                                  #Access,
                                  0);

if dRC = 0 then

// die zweiten erweiterten Zugriffsrechte für den FTP-Benutzer „FTPUser“ abrufen
dRC = ocFTPSvr.GetUserAccessInfo("FTPUser",
                                  #Path[0],
                                  sizeof(Path),
                                  #Access,
                                  0);

// ... bis GetUserAccessInfo FTPSVR_ERR_NOT_FOUND zurückliefert

end_if;
```

7.6.1.8 GetUserExtendedInfo

Dieser Befehl ruft erweiterte Informationen von einem bestimmten FTP-Benutzer ab.

```
FUNCTION __cdecl virtual global GetUserExtendedInfo
VAR_INPUT
  Benutzername  : ^CHAR;
  pValue        : pVoid;
  SizeOfValue   : UDINT;
  UserInfo      : UDINT;
END_VAR
VAR_OUTPUT
  dRC (EAX)    : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
Benutzername	^CHAR	0-terminierter String, der den Namen des FTP-Benutzers enthält, von dem die erweiterten Informationen abgerufen werden

pValue	pVoid	Zeiger auf die Variable, die den Wert der erweiterten Informationen abruft.
SizeOfValue	UDINT	Größe der Variablen, Array oder des Speichers in Byte, auf den pValuepoints zeigt.
UserInfo	UDINT	die Art der erweiterten Informationen abrufen FTPS_USR_INFO_REFERENCE FTPS_USR_INFO_STATUS FTPS_USR_INFO_LOGGEDIN
Rückgabeparameter	Typ	Beschreibung
dRC	DINT	0 Erfolg <0 Negativer Fehler-Code

Die folgende Tabelle zeigt die verfügbaren erweiterten Informationen sowie den Datentyp und die Größe des Parameters pValue.

UserInfo	Typ	Size	DLL Version
FTPS_USR_INFO_REFERENCE	USINT	1	01.01.001
FTPS_USR_INFO_STATUS	DINT	4	01.01.001
FTPS_USR_INFO_LOGGEDIN	DINT	4	01.01.001

Siehe [Anhang A - Typen](#) für die Werte und die Bedeutung für alle erweiterte Informationen.

Beispiel

```

dRC : DINT
ref : USINT;
stat : UDINT;

// Erweiterte Info "Reference" abrufen
dRC := ocFTPSvr.GetUserExtendedInfo("FTPUser",
                                     #ref,
                                     sizeof(ref),
                                     FTPS_USR_INFO_REFERENCE);

// Erweiterte Info „Status“ abrufen
dRC := ocFTPSvr.GetUserExtendedInfo("FTPUser",
                                     #ref,
                                     sizeof(ref),
                                     FTPS_USR_INFO_STATUS);

```

7.6.1.9 GetUserInfo

Diese Methode wird zum Abfragen aller registrierten FTP-Benutzer, ihrer allgemeinen Zugriffsrechten und des Hauptverzeichnisses verwendet.

```
FUNCTION __CDECL VIRTUAL GLOBAL GetUserInfo
VAR_INPUT
    Benutzername      : ^CHAR;
    SizeOfUsername   : UDINT;
    GeneralAccess    : ^UDINT;
    RootDir          : ^CHAR;
    SizeOfRootDir    : UDINT;
    UserIdx          : UDINT;
END_VAR
VAR_OUTPUT
    dRC (EAX)        : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
Benutzername	^CHAR	Zeiger auf einen Array oder zugeteilten Speicherbereich, der den Namen des FTP-Benutzers abruft.	
SizeOfUsername :	UDINT	Größe des Arrays oder Speichers in Byte, auf den der Benutzername zeigt.	
GeneralAccess	^UDINT	Zeiger auf eine Variable vom Typ UDINT, um den Wert des Allgemeinzugangs zu erhalten.	
RootDir	^CHAR	Zeiger auf einen Array oder Speicherbereich, um das Hauptverzeichnis des FTP-Benutzers abzurufen.	
SizeOfRootDir	UDINT	Größe des Arrays oder Speichers in Byte auf den das RootDir zeigt	
UserIdx	UDINT	Index des FTP-Benutzers zum Abrufen (Beginn mit 0)	
Übergabeparameter	Typ	Beschreibung	
dRC	DINT	0 FTP Benutzer wurde erfolgreich gefunden FTPSVR_ERR_NOT_FOUND Kein FTP-Benutzer verfügbar NOT_FOUND	FTP Benutzer wurde erfolgreich gefunden Kein FTP-Benutzer verfügbar NOT_FOUND

Um die Informationen aller FTP-Benutzer abzurufen, muss die Methode aufgerufen werden, bis `FTPSVR_ERR_NOT_FOUND` zurückgeliefert wird. Mit jedem Aufruf muss der Parameter `UserIdx` sich um eins erhöhen. Ein `UserIdx` von 0 liefert die Informationen des ersten FTP-Benutzer zurück.

Beispiel

```

Benutzername  : ARRAY [0..FTPSVR_USERNAME_LEN] OF CHAR
RootDir       : ARRAY [0..260] OF CHAR;
GeneralAccess : UDINT;
dRC          : DINT;

// first user
dRC := ocFTPSvr.GetUserInfo(#Username[0],
                           sizeof(Username),
                           #GeneralAccess,
                           #RootDir[0],
                           sizeof(RootDir),
                           0);

if dRC = 0 then

// zweite User
dRC := ocFTPSvr.GetUserInfo(#Username[0],
                           sizeof(Username),
                           #GeneralAccess,
                           #RootDir[0],
                           sizeof(RootDir),
                           0);

// ... bis GetUserInfo FTPSVR_ERR_NOT_FOUND zurückliefert

end_if;

```

7.6.1.10 RemoveAccess

Diese Anweisung entfernt eine bestehende erweiterte Zugangsberechtigung.

```

FUNCTION __CDECL VIRTUAL GLOBAL RemoveAccess
VAR_INPUT
    Benutzername  : ^CHAR;
    Path         : ^CHAR;
END_VAR
VAR_OUTPUT
    dRC (EAX)    : DINT;
END_VAR;

```

Übergabeparameter	Typ	Beschreibung
Benutzername	^CHAR	0-terminierter String mit dem Namen des FTP-Benutzers, dessen Zugangsberechtigung entfernt werden soll
Path	^CHAR	0-terminierter String, der den Pfad der zu entfernende Zugangsberechtigung enthält
Rückgabeparameter	Typ	Beschreibung

dRC	DINT	0 Erfolg
		<0 Negativer Fehler-Code

Es ist nicht möglich ein bestehendes Zugriffsrecht zu entfernen, wenn ein FTP Client mit dem gleichen Namen wie der FTP Benutzer angemeldet ist, dessen Zugriffsrechte entfernt werden soll.

Ist der FTP-Benutzer mit der Methode [AddUser\(\)](#) eingefügt und der Parameter Save auf 1 gesetzt, ist es möglich das Zugriffsrecht in der Datei ftps.ini durch den Aufruf der Methode [UpdateFTPSini\(\)](#) zu entfernen.

Beispiel

```
dRC : DINT;

dRC := ocFTPSvr.RemoveAccess("FTPUser", "C:\TEMP");
if dRC = 0 then
// Zugriffsrechte erfolgreich eingefügt
else
// Fehler aufgetreten
end_if
```

7.6.1.11 RemoveUser

Entfernt einen bestehenden FTP-Benutzer.

```
FUNCTION __cdecl virtual global RemoveUser
VAR_INPUT
    Benutzername : ^CHAR;
END_VAR
VAR_OUTPUT
    dRC (eax) : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
Benutzername		0-terminierter String mit dem zu entfernenden Benutzernamen	
Rückgabeparameter	Typ	Beschreibung	
dRC	DINT	0	Erfolg
		<0	Negativer Fehler-Code

Es ist nicht möglich einen FTP-Benutzer zu ändern, wenn ein FTP-Client mit dem Namen des FTP-Benutzereintrags, der entfernt werden soll, eingeloggt ist.

Diese Methode entfernt nicht die FTP-Benutzer von der Datei `ftps.ini`.

Zum Aktualisieren der Datei `ftps.ini` muss die Methode [UpdateFTPSini\(\)](#) aufgerufen werden

Beispiel

```
dRC : DINT;

// der Benutzername eines bestehenden Eintrags ändern
dRC := ocFTPSvr.RemoveUser("NewFTPUser");
if dRC = 0 then
// FTP erfolgreich geändert
else
// Fehler aufgetreten
end_if;
```

7.6.1.12 StartServer

Diese Methode startet den FTP-Server, falls dieser nicht bereits läuft.

```
FUNCTION __CDECL VIRTUAL GLOBAL StartServer
VAR_OUTPUT
  dRC (EAX) : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung
keine		
Rückgabeparameter	Typ	Beschreibung
dRC	DINT	0 Erfolg <0 Negativer Fehler-Code

Wenn der FTP-Server bereits läuft, wird `FTPSVR_ERR_EXISTS` zurückgegeben.

Die Methode liest die Daten aus der Datei `ftps.ini` und erstellt die FTP-Benutzer, die in der Datei gespeichert sind.

Die benötigte Zeit der Methode hängt von der Größe der `ftps.ini` Datei ab. Die Methode sollte daher bei zyklischen oder Echtzeit-Aufgaben nicht aufgerufen werden.

Beispiel

```
dRC : DINT;

dRC := ocFTPSvr.StartServer();
if dRC = 0 then
```

```
// start server fertig
else
// start server fehlgeschlagen
end_if
```

7.6.1.13 StopServer

Stoppt den FTP-Server und trennt alle aktiv verbundenen und eingeloggten FTP-Clients.

```
FUNCTION __CDECL VIRTUAL GLOBAL StopServer
VAR_OUTPUT
    dRC (EAX) : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
keine			
Rückgabeparameter	Typ	Beschreibung	
dRC	DINT	0	Erfolg
		<0	Negativer Fehler-Code

Alle erstellten FTP-Benutzer die nicht in der Datei FTPS.INI gespeichert sind, gehen beim Aufruf dieser Methode verloren. Alle aktiv verbundenen und eingeloggten Clients werden vom Server getrennt, egal ob die Datenübertragung abgeschlossen ist oder nicht. Die benötigte Zeit der Methode hängt von der Anzahl der aktiven verbunden Clients ab die getrennt werden müssen und der Zahl der registrierten FTP-Benutzer. Die Methode sollte daher bei zyklischen oder Echtzeit-Aufgaben nicht aufgerufen werden.

Beispiel

```
dRC : DINT;

dRC := ocFTPSvr.StopServer();
if dRC = 0 then
// Stopp Server fertig
else
// Stopp Server fehlgeschlagen
end_if;
```

7.6.1.14 SetConfig

Dieser Befehl wird benutzt, um die aktuelle Konfiguration des FTP-Servers zu ändern.

```
FUNCTION __CDECL VIRTUAL GLOBAL SetConfig
VAR_INPUT
    pValue      : ^CHAR;
    udValue     : UDINT;
    Option      : UDINT;
END_VAR
VAR_OUTPUT
    dRC (EAX) : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung				
pValue	^CHAR	0-terminierter String der den Wert der zu einzustellenden Option enthält				
udValue	UDINT	Variable die den Wert der zu einzustellenden Option enthält				
Option	UDINT	Die zu einstellenden Optionen <ul style="list-style-type: none"> • FTPS_OPT_READ_TMO • FTPS_OPT_WRITE_TMO • FTPS_OPT_CMD_TMO • FTPS_OPT_NO_DRIVE • FTPS_OPT_BACKSLASH • FTPS_OPT_FORWARDSLASH • FTPS_OPT_MAX_SESSIONS • FTPS_OPT_DEF_ROOT_DIR • FTPS_OPT_PASV_PORT • FTPS_OPT_ANONYMOUS • FTPS_OPT_ANONYMACCESS • FTPS_OPT_NO_FTPSINI • FTPS_OPT_SVR_D_PRIO • FTPS_OPT_SVR_T_PRIO • FTPS_OPT_WINCOMP_LISTFORMAT 				
Rückgabeparameter	Typ	Beschreibung				
dRC	DINT	<table border="1"> <tr> <td>0</td><td>Erfolg</td></tr> <tr> <td><0</td><td>Negativer Fehler-Code</td></tr> </table>	0	Erfolg	<0	Negativer Fehler-Code
0	Erfolg					
<0	Negativer Fehler-Code					

Wird pValue Parameter verwendet, wird der Parameter udValue ignoriert.

Für verfügbare Optionen und ihre möglichen Werte siehe „Kapitel Server Konfiguration“. Alle Werte in Anführungszeichen (z.B. "ON", "OFF", des Standard-Hauptverzeichnis) können mit dem Parameter pValue verwendet werden. Alle anderen müssen mit dem Parameter udValue benutzt werden, um die angegebene Option zu setzen. Wird der udValue Parameter verwendet, muss der Parameter pValue ein NIL-Zeiger 0 sein.

Beispiel

```
dRC : DINT;

// ANONYM Option setzen(anonymous FTP)

// Parameter pValue verwenden
dRC := ocFTPSvr.SetConfig("ON", 0, FTPS_OPT_ANONYMOUS);

// Parameter udValue verwenden
dRC := ocFTPSvr.SetConfig(NIL, 1, FTPS_OPT_ANONYMOUS);

// Standard Hauptverzeichnis setzen
dRC := ocFTPSvr.SetConfig("C:\FTPROOT", 0, FTPS_OPT_DEF_ROOT_DIR);
```

7.6.1.15 UpdateFTPSini

Diese Anweisung schreibt alle FTP-Benutzer und deren erweiterte Zugriffsrechte auf die Datei `ftps.ini`.

```
FUNCTION __CDECL VIRTUAL GLOBAL UpdateFTPSini
VAR_INPUT
    non_blocking : UDINT;
END_VAR
VAR_OUTPUT
    dRC (EAX)      : DINT;
END_VAR;
```

Übergabeparameter	Typ	Beschreibung	
non_blocking	UDINT	0	blocking mode
		1 non-blocking mode	
Ist der Non-Blocking Modus eingestellt, schreibt ein interner Betriebssystemtask die Daten in die <code>ftps.ini</code> Datei.			
Rückgabeparameter	Typ	Beschreibung	
dRC	DINT	0	Erfolg

		<0 Negativer Fehler-Code
--	--	--------------------------

Mit dieser Methode werden Daten in eine Datei geschrieben. Die benötigte Laufzeit dieser Methode hängt von der Größe der in die Datei zu schreibenden Daten ab. Je mehr verfügbare FTP-Benutzer, desto mehr Zeit wird benötigt um sie in der Datei `ftps.ini` zu speichern. Diese Methode blockiert alle anderen Anweisungen, bis alle Daten auf die Festplatte geschrieben worden sind.

Es ist möglich die Methode im Non-Blocking Modus aufzurufen. In diesem Fall schreibt ein interner Betriebssystemtask die Daten. Die Methode [GetUpdateFTPSiniStatus\(\)](#) kann verwendet werden, um den Status des Schreibprozesses im Non-Blocking Modus abzurufen.

Sollen mehr FTP-Benutzer bzw. Zugangsberechtigungen eingefügt, geändert oder entfernt werden, soll diese Methode nach der Änderung, Einfügung oder Entfernung aller FTP-Nutzer aufgerufen werden.

Beispiel

```
// add FTP user 1, 2, 3,..., remove another, edit another
// add access..., ...

dRC : DINT;

dRC := ocFTPSvr.UpdateFTPSini(0);
if dRC = 0 then
// FTP-Benutzer und Zugriffsrechten ins ftps.ini geschrieben
else
// Fehler aufgetreten
end_if;
```

7.6.2 Quick Review

Die benötigte Zeit von den Methoden der `_FTPServer` Klasse kann auf Grund äußerer Faktoren nicht berechnet werden (insbesondere File Access). Es wird nicht empfohlen, diese Funktion von zyklischen oder Echtzeit Tasks aufzurufen.

7.6.3 `FTPS.INI`

Wird diese Methode mit den richtigen Werten der entsprechenden Parameter aufgerufen, werden alle FTP-Benutzer und Zugangsrechte, die mittels der CLI und OS-Interface Library der `_FTPServer` Klasse eingefügt wurden, in die Datei `ftps.ini` gespeichert.

Die Datei `ftps.ini` ist eine Textdatei, die mit einem beliebigen Text-Editor bearbeitet werden kann.

Inhaltsformat der FTPS.INI Datei

U	benutzername
P	encoded password
GA	general access
R	root directory
A	erweiterte Zugriffsrechte, Pfad Zugang, path2 access2

Beispiel

```
[U=FTPUser]
P=X@$
GA=R
R=C:\TEMP\
A=C:\TEMP|W
A=D:\|N
```

... Nächster Benutzer ...



Nach dem letzten Benutzereintrag in der `ftps.ini` Datei ist ein „newline und „carriage return“ (\r\n) erforderlich. Ansonsten kann die Datei nicht aus dem OS LASAL FTP-Server gelesen werden.

Die `FTPS.INI` Datei wird in das Hauptverzeichnis C:\ (C:\ftps.ini) gespeichert.

Wird die `ftps.ini` Datei in einen IPC oder eine CPU transferiert, werden die FTP-Benutzer in die Datei beim nächsten Start des FTP-Servers geladen.

Das Passwort in der Datei ist ein codiertes Passwort. Zum Codieren eines Passworts verwenden sie das Windows-Programm PwdGen.exe. Ist der Inhalt der Datei für einen bestimmten Benutzer nicht gültig, wird es nicht gespeichert und beim registrierten Nutzer beim Start hinzugefügt. Ist der komplette Inhalt der Datei ungültig, wird kein FTP-Benutzer geladen.

7.6.4 Fehler-, Warnung- und Info-Protokollierung

Mit der Command Line Interface (CLI) Anweisung, SET DBGLEVEL FTPSERVER, kann die Informationsprotokollierung aktiviert werden.

DBGLEVEL FTPSERVER

0	keine Protokollierung
1	Nur Fehlermeldungen
2	Fehler- und Debug-Meldungen
3	Erweiterte Informationen

Ist die Standardfehler-Protokollierung 1, so werden nur Fehlermeldungen gesendet.

Fehlerprotokollierung kann beim Start über die autoexec.lsl oder manuell durch das Command Line Interface (CLI) Befehl aktiviert werden.

Beispiel

```
SET DBGLEVEL FTPSERVER 2
```

Für die Fehlerprotokollierung und Debug-Meldungen

Der FTP-Server erzeugt immer den folgenden erweiterte Informationseintrag beim Start:

```
*timestamp*;FTPSVR;3;Initializing FTP-Server interface *dll version*
```

7.7 Anhang

7.7.1 Typen

Typen für Zugriffsrechte

Zugriff	Wert	DLL Ver.	Beschreibung
FTP_ACCESS_NONE	0	01.01.001	Kein Zugang
FTP_ACCESS_READ	1	01.01.001	Nur Lesezugriff
FTP_ACCESS_WRITE	2	01.01.001	Voller Zugriff (lesen / schreiben / löschen)

Referenztypen

Referenz	Wert	DLL Ver.	Beschreibung

FTPS_USR_REF_OS	1	01.01.001	Eintrag wurde durch den FTP-Server eingefügt
FTPS_USR_REF_APPL	3	01.01.001	Eintrag wurde mittels der Methode AddUser eingefügt und mit einem Wert von 0 für den Parameter Save

Statustypen

Status	Wert	DLL Ver.	Beschreibung
FTPS_USR_STAT_ADDED	0x00000001	01.01.001	Eintrag eingefügt, nicht ins ftps.ini gespeichert
FTPS_USR_STAT_CHANGED	0x00000002	01.01.001	Eintrag geändert
FTPS_USR_STAT_STORED	0x00000004	01.01.001	Eintrag in der Datei ftps.ini gespeichert

Typen für Konfigurationsoptionen

Option	Wert	DLL Ver.	Beschreibung
FTPS_OPT_READ_TMO	2	01.01.001	Lesen Timeout (empfangen)
FTPS_OPT_WRITE_TMO	3	01.01.001	Schreiben Timeout (senden)
FTPS_OPT_CMD_TMO	4	01.01.001	Command Timeout (empfangen)
FTPS_OPT_NO_DRIVE	5	01.01.001	Kein Laufwerk auf PWD
FTPS_OPT_BACKSLASH	6	01.01.001	Wechseln auf Backslash (empfangen)
FTPS_OPT_FORWARDSLASH	7	01.01.001	Wechseln auf Schrägstrich (senden)
FTPS_OPT_MAX_SESSIONS	8	01.01.001	Anzahl der max. eingeloggten FTP-Clients
FTPS_OPT_DEF_ROOT_DIR	9	01.01.001	Standard Hauptverzeichnis des FTP-Servers
FTPS_OPT_PASV_PORT	10	01.01.001	Port Zahl der passiven Datenverbindung
FTPS_OPT_ANONYMOUS	11	01.01.001	Anonym FTP
FTPS_OPT_ANONYMACCESS	12	01.01.001	Anonym FTP Zugang
FTPS_OPT_NO_FTPSINI	13	01.01.001	Keine ftps.ini Datei
FTPS_OPT_SVR_D_PRIO	14	01.01.001	Priorität der FTP-Server Daemon Task

FTPS_OPT_SVR_T_PRIO	15	01.01.001	Priorität der einzelnen Verbindungen (FTP-Client)
FTPS_OPT_WINCOMP_LISTFORMAT	16	01.01.004	Windows-kompatible Listformat

Typen für erweiterte Benutzerinformationen

UserInfo	Wert	DLL Ver.	Beschreibung
FTPS_USR_INFO_REFERENCE	1	01.01.001	Referenz des Eintrags, siehe Referenztypen
FTPS_USR_INFO_STATUS	2	01.01.001	Status eines Eintrags, siehe Statustypen
FTPS_USR_INFO_LOGGEDIN	3	01.01.001	Anzahl der eingeloggten FTP-Benutzer

Weitere Typen

Typ	Wert	DLL Ver.	Beschreibung
FTPSVR_INF	0xFFFF	01.01.001	Timeout Wert (unbegrenzt)
FTPSVR_USERNAME_LEN	63	01.01.001	Max. Benutzernamenlänge
FTPSVR_PASSWORD_LEN	63	01.01.001	Max. Länge des Passwortes

7.7.2 Fehlerwerte

Fehler	Wert	DLL Ver.	Beschreibung
FTPSVR_ERR_VF	-1000	01.01.001	Fehler beim Initialisieren des internen virtuellen Dateisystems
FTPSVR_ERR_SEMA	-1001	01.01.001	Fehler bei der Erstellung eines Flags
FTPSVR_ERR_DEAMON	-1002	01.01.001	Fehler bei der Erstellung eines FTP-Server Daemon Threads
FTPSVR_ERR_POINTER	-1003	01.01.001	Ungültiger Zeiger, NIL Fehler
FTPSVR_ERR_PARAM	-1004	01.01.001	Ungültige Parameter (Benutzername zu lang, ...)
FTPSVR_ERR_NOMEM	-1005	01.01.001	Nicht ausreichend Speicher
FTPSVR_ERR_NOT_FOUND	-1006	01.01.001	Eintrag nicht gefunden (Benutzer, Zugang, ...)

FTPSVR_ERR_EXISTS	-1007	01.01.001	Eintrag existiert bereits (Benutzer, Zugang, ...)
FTPSVR_ERR_INI_OPEN	-1008	01.01.001	Fehler beim Erstellen / Öffnen einer ftps.ini Datei
FTPSVR_ERR_INI_WRITE	-1009	01.01.001	Fehler beim Schreiben in der Datei ftps.ini
FTPSVR_ERR_INI_READ	-1010	01.01.001	Fehler beim Lesen von Datei ftps.ini
FTPSVR_ERR_UPDATETASK	-1011	01.01.001	Update Task konnte nicht gestartet werden
FTPSVR_ERR_UPD_BUSY	-1012	01.01.001	Update Task beschäftigt
FTPSVR_ERR_UPD_NO_TASK	-1013	01.01.001	Update Task nicht gestartet
FTPSVR_ERR_LOGGED_IN	-1014	01.01.001	Nutzer eingeloggt
FTPSVR_ERR_NO_INTERFACE	-1015	01.01.001	Fehler, keine FTP-Server-Schnittstelle vorhanden

7.7.3 Flags

Flag zur Änderung eines FTP-Benutzers (EditUser)

Flag	Wert	DLL Ver.	Beschreibung
FTPSVR_CHANGE_PASSWORD	0x00000001	01.01.001	Passwort ändern
FTPSVR_CHANGE_GENERALACCESS	0x00000002	01.01.001	Allgemeinen Zugang ändern
FTPSVR_CHANGE_ROOTDIR	0x00000004	01.01.001	Hauptverzeichnis ändern

8 Log-Datei

8.1 Allgemein

Dieses Dokument erklärt die Einträge der EVENT00.LOG-Log-Datei. Des weiteren sind auch mögliche Fehlerursachen und Lösungsvorschläge beschrieben. In diesem Dokument sind die wichtigsten Einträge dokumentiert.

Die meisten Log-Datei-Einträge beginnen mit Datum und Uhrzeit.

Datums-Format ist Jahr/Monat/Tag z.B.: 08/11/29; .

Uhrzeit-Format ist Stunden:Minuten:Sekunden.Millisekunden z.B.: 13:49:26.347;

Manche Einträge in der Log-Datei beginnen mit *I*6E: . Das „*I“ weist darauf hin, dass der Eintrag aus einem Hardware-Interrupt erfolgte. Die folgende Nummer ist ein laufender Zähler in hexadezimal.

Einträge am Ende der Log-Datei sind immer die aktuellsten.

8.2 Detaillierte Aufschlüsselung einzelner Logeinträge

8.2.1 System

Eintrag	PowerON,OS:01.02.035(Oct 15 2008,15:22:20),ETV 0811,PLC-SerNbr.01973776,FPGA V14 Oder ----- PowerON,OS:01.02.053(Apr 21 2009,10:44:57),CIPC,PLC-SerNbr.01163204,FPGA V23 -----
Beschreibung	System wurde eingeschaltet. Version des SPS-Betriebssystems. 01.02.035 Releasedatum: 15.Oktober 2008 Releaseuhrzeit: 15:22:20 CPU Typ: ETV0811 Seriennummer: 01973776 FPGA Version: 1.4 (FPGA-Versionsnummer nicht immer vorhanden)
Ursache / Abhilfe	System wurde eingeschaltet.
Eintrag	DLL_LoadLib: Loading library "rexx.dll" failed, rc=2
Beschreibung	Das Laden einer DLL (hier „rexx.dll“) ist fehlgeschlagen. „rc“ ist der Rückgabewert der Funktion LoadLibrary.
Ursache / Abhilfe	Die Datei (hier „rexx.dll“) fehlt im System oder ist an einer falschen Stelle, so dass sie nicht gefunden wird.
Eintrag	runStatus changed from 0 to 26

	Oder runStatus changed from RUN RAM(0) to WAIT(26)
Beschreibung	CPU-Status der Steuerung hat sich verändert CPU-Status änderte sich von 0 auf 26 Die Zahlen müssen mit der Tabelle für CPU-Status interpretiert werden
Ursache / Abhilfe	Steuerungsprogramm startet oder wird beendet Genaueres siehe Kapitel der CPU-Status
Eintrag	PrjName:Class,Chksum:0xC3492D3D
Beschreibung	Applikation mit Namen „Class“ und Checksumme 0xC3492D3D wird gestartet. Wenn sich die Checksumme seit dem letzten Eintrag geändert hat wurde eine andere Softwareversion eingespielt (Update oder Entwicklungsstadium).
Ursache / Abhilfe	Info
Eintrag	SFB on (00388264)
Beschreibung	ShadowFrameBuffer wurde aktiviert. Grafikbefehle zeichnen von nun an in einem Speicherbereich, der nicht sichtbar ist.
Ursache / Abhilfe	Info
Eintrag	SFB off (004225F0)
Beschreibung	ShadowFrameBuffer wurde aktiviert. Grafikbefehle zeichnen in dem Speicherbereich, der zur Zeit sichtbar ist.
Ursache / Abhilfe	Info
Eintrag	*I*9A:Power fail! Oder USV Power-Down erkannt
Beschreibung	Es wurde eine Spannungsunterbrechung festgestellt.
Ursache / Abhilfe	Steuerung wird ausgeschaltet
Eintrag	MakeBootDisk (ONLINE): filename = C:\IPC.RTB MakeBootDisk (ONLINE): result = 0
Beschreibung	Betriebssystem Update wurde vom Anwender durchgeführt

Ursache / Abhilfe	Info
Eintrag	Reboot
Beschreibung	Neustart wurde ausgeführt
Ursache / Abhilfe	Info
Eintrag	WATCHDOG Error!
Beschreibung	Der Watchdog wurde 130 ms nicht getriggert. Die Peripherie (DIAS-Bus / VARAN-Bus / C-DIAS) geht in Fail Safe Zustand.
Ursache / Abhilfe	<p>Der Realtime Task wurde mindestens 30 ms blockiert. Tritt daher meist in Kombination mit einem Realtime Runtime Fehler auf.</p> <p>Bei Eintrag direkt nach der Initialisierungsphase: DIAS-Bus und/oder VARAN-Bus nicht verwendet (kein Realtime Runtime Fehler)</p>
Eintrag	<p>Exception 0xC0000094, EIP=0x00E4D164</p> <p>Class:CpStat=24</p> <p>EXCEPTION - Dump:</p> <pre> EIP:00E4D164 EAX:02FAF080 EBX:0007A136 ECX:FFFFFF EDX:00000000 ESI:01904344 EDI:00000000 ESP:0E82FDB0 EBP:0E82FDC4 EFL:00003216 CS:0000001B DS:01940023 SS:01940023 ES:01940023 FS:00000000 GS:00000000 Err in Class @ .\Class\OhmMeter\OhmMeter.st::_READ@OHMMETER (000327) ESI's Objname: PUFFERLADR__BASE\OHMEINGANG TASK=WSP0__CT,STACK=0E80FD90-0E82FE90,MIN=126672d APPHEAP:Start=017E3528,Size=217893592d,Used=1692268d,Free=216201324d USRCODE:Start=00DC0000,Size=10485760d USRDATA:Start=017C0000,Size=218038272d STACK: 0E82FD70:23 00 94 01 1B 00 82 0E 16 32 00 00 00 00 00 00 #.....2..... 0E82FD80:44 43 90 01 C4 FD 82 0E 9C FD 82 0E 36 A1 07 00 DC.....6... 0E82FD90:00 00 00 00 FF FF FF 80 F0 FA 02 64 D1 E4 00d... 0E82FDA0:94 00 00 C0 00 00 00 00 00 00 00 00 00 00 00 00 0E82FDB0:00 00 00 00 00 00 80 F0 FA 02 44 43 90 01DC.... 0E82FDC0:E5 C0 E4 00 D4 FD 82 0E 24 DF E4 00 CC 47 90 01\$....G.... 0E82FDD0:00 00 00 00 30 FE 82 0E 48 B3 26 00 F0 1A 80 0E0...H.&..... 0E82FDE0:00 00 7C 01 00 4C 7F 0E 30 FE 82 0E 00 FE 82 0EL..0.....</pre>

0E82FDF0:00 00 00 00 03 18 00 00 FF
0E82FE00:00 00 7C 01 00 4C 7F 0E 00 00 00 00 94 00 00 C0 ..|..L.....
0E82FE10:DC FD 82 0E 50 1B 80 0E 00 FE 82 0E F4 FB 82 0EP.....
0E82FE20:FF FF FF DC 9B 2C 00 C8 4D 36 00 00 00 00 00M6.....
0E82FE30:C8 C1 89 0E 71 D3 23 00 30 70 89 0E 03 18 00 00q.#.Op.....
0E82FE40:1C C2 89 0E 03 18 00 00 F8 C1 89 0E 03 18 00 00
0E82FE50:16 00 00 00 17 00 00 00 00 00 00 00 00 00 4C 7F 0EL..
0E82FE60:01 00 00 00 00 00 00 83 D0 23 00 00 4C 7F 0E#.L..
0E82FE70:00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0E82FE80:01 00 00 00 3D 55 20 00 00 4C 7F 0E 00 00 00 00=U ..L.....
0E82FE90:60 05 80 0E 50 06 80 0E 11 20 00 00 35 53 3D 43 `...P.... ..5S=C
0E82FEA0:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FEB0:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FEC0:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSS
SSSSSSSSSSSSSSSS
0E82FED0:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSS
SSSSSSSSSSSSSSSS
0E82FEE0:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FEF0:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FF00:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FF10:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FF20:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FF30:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FF40:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FF50:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS
0E82FF60:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSSSSSS

Main Task :0010000-001BFF88,001BFDOC,780160d
Idle Task :000CCDA0-000CCFA0,000CCF60,356d
Main Task Low :000D5610-000D6310,000D62AC,336d
Main Task High :000D6720-000D7020,000D6F60,356d
CPU Monitor :000D8040-000D8240,000D81C8,300d
DLED_Task :000D8630-000D8B30,000D8ADC,1068d
RT_AsyncTask :000DA280-000DB380,000DB2E8,4140d
UserLogTask :0E7B0010-0E7B2110,0E7B1F58,4820d
CanRxMailThread :0E7B4C60-0E7B5D60,0E7B5C74,4112d
VARAN Task :0E7BC3D0-0E7C44D0,0E7C41C8,30568d
EMAC-Task :0E7C4AF0-0E7C5BF0,0E7C5B30,4116d
IPTASK :0E7DF280-0E7E0380,0E7E02BC,3644d

```
INTERRUPT  :0E7E0CE0-0E7E1DE0,0E7E1D1C,4064d
IPTASK    :0E7E2320-0E7E3420,0E7E335C,4116d
INTERRUPT  :0E7E3980-0E7E4A80,0E7E49BC,4116d
IPTASK    :0E7E4FE0-0E7E60E0,0E7E601C,4116d
INTERRUPT  :0E7E6640-0E7E7740,0E7E767C,4116d
TIMER     :0E7E7D30-0E7E8E30,0E7E8D68,3676d
TCP Server :0E7E9860-0E7EB960,0E7EB430,6468d
USB Iso Task :0E7EBE90-0E7EC790,0E7EC6DC,2068d
USB Hub   :0E7ED7E0-0E7EE8E0,0E7EE81C,3112d
DbgRxFast :0E899F30-0E89C030,0E89BF00,8060d
WSP0_CT   :0E80FD90-0E82FE90,0E82FAAC,126672d
TCP Client 1 :0E8035B0-0E8076B0,0E8066E8,12272d
WSP0_RT   :0E89C790-0E8BC890,0E8BC804,130152d
WSP0_BT   :0E82FEA0-0E84FFA0,0E84FF48,126528d
WSP0_LT   :0E8BD460-0E8BF560,0E8BF544,8420d
```

CALL STACK:

```
0E82FDB0: Class @.\Class\OhmMeter\OhmMeter.st::_READ@OHMMETER
(000327)
0E82FDD8:EIP=0x0026B348
0E82FE34:EIP=0x0023D371
0E89C1CC:EIP=0x00000000
```

oder

Exception ACCESS EXCEPTION(0xC0000005), EIP=0x00E36243

09/05/13:08:01:23.457;VTI011:CpStat=ACCESS EXCEPTION(50)

EXCEPTION - Dump:

```
EIP:00E36243 EAX:00000006 EBX:019191C0 ECX:00000003
EDX:00000002 ESI:01918878 EDI:00000000 ESP:0387DC94
EBP:0387E094 EFL:00003256 CS:0000001B DS:00000023
SS:00000023 ES:00000023 FS:00000000 GS:00000000
```

Err in VTI011@.\Class\CheckIOKopf\CheckIOKopf.st::_CYWORK@CHECKIOKOPF
(001365)

ESI's Objname: CHECKIOKOPF1

TASK=WSP0_CT,STACK=0385E050-0387E150,MIN=127324d

APPHEAP:Start=017CF3DC,Size=32705572d,Used=1161912d,Free=31543660d

USRCODE:Start=00DC0000,Size=10485760d

USRDATA:Start=017C0000,Size=32768000d

STACK:

0387DC54:23 00 00 00 1B 00 00 00 56 32 00 00 00 00 00 00 00 #.....V2.....
0387DC64:78 88 91 01 94 E0 87 03 80 DC 87 03 C0 91 91 01 x.....
0387DC74:02 00 00 00 03 00 00 06 00 00 00 43 62 E3 00Cb..
0387DC84:05 00 00 C0 00 00 00 02 00 00 00 74 A6 3C 00t.<.
0387DC94:FF FF FF FF 06 00 00 00 78 88 91 01 00 00 00 00x.....
0387DCA4:FF FF
0387DCB4:FF FF
0387DCC4:FF FF
0387DCD4:FF FF
0387DCE4:FF FF
0387DCF4:FF FF
0387DD04:FF FF
0387DD14:FF FF
0387DD24:FF FF
0387DD34:FF FF
0387DD44:FF FF
0387DD54:FF FF
0387DD64:FF FF
0387DD74:FF FF
0387DD84:FF FF
0387DD94:FF FF
0387DDA4:FF FF
0387DDB4:FF FF
0387DDC4:FF FF
0387DDD4:FF FF
0387DDE4:FF FF
0387DDF4:FF FF
0387DE04:FF FF
0387DE14:FF FF
0387DE24:FF FF
0387DE34:FF FF
0387DE44:FF FF

Main Task :00100000-001BFF88,001BFDOC,780160d

Idle Task :000CCDA0-000CCFA0,000CCF60,356d

Main Task Low :000D5640-000D6340,000D62DC,432d

Main Task High :000D6750-000D7050,000D6F90,324d

CPU Monitor :000D8070-000D8270,000D81FC,300d

```

DLED_Task      :000D8660-000D8B60,000D8B0C,1068d
RT_AsyncTask   :000DA2B0-000DB3B0,000DB318,4140d
UserLogTask    :03700010-03702110,03701F58,4872d
CanRxMailThread :03704D10-03705E10,03705D24,4112d
VARAN Task     :037F7350-037FF450,037FF150,30692d
EMAC-Task      :037FFAE0-03800BE0,03800B20,4140d
IPTASK         :0381AFF0-0381C0F0,0381C02C,3724d
INTERRUPT      :0381C100-0381D200,0381D13C,4116d
IPTASK         :0381D350-0381E450,0381E38C,4116d
INTERRUPT      :0381E9B0-0381FAB0,0381F9EC,4116d
IPTASK         :03820010-03821110,0382104C,4116d
INTERRUPT      :03821670-03822770,038226AC,4116d
TIMER          :03822D60-03823E60,03823D98,3676d
TCP Server     :038248A0-038269A0,03826470,6468d
USB Iso Task   :03826ED0-038277D0,0382771C,2068d
USB Hub         :0382BEC0-0382CFC0,0382CEE0,3112d
DbgRxFast      :038DF090-038E1190,038E1060,8108d
TCP Client 1   :03847B40-0384BC40,0384AC78,12272d
WSP0_CT         :0385E050-0387E150,0387D990,127324d
WSP0_RT         :038EDFF0-0390E0F0,0390E064,130096d
WSP0_LT         :03837990-03839A90,03839A74,8420d

```

CALL STACK:

0387DC94:VTI011@.

\Class\CheckIOKopf\CheckIOKopf.st::_CYWORK@CHECKIOKOPF (001365)

0387E098:EIP=0x0026CBF8

0387E0F4:EIP=0x0023D7D1

0383A2CC:EIP=0x00000000

Beschreibung

Exception mit Exceptioncode und Instruction Pointer (EIP)

Mit dem Dump kann man offline herausfinden wo in der Applikation der Fehler auftritt.

Exceptioncodes mit Bedeutung:

0xC0000005 : "ACCESS EXCEPTION"

Es wurde auf eine logische Speicheradresse zugegriffen, die nicht existiert.
z.B. Speicherzugriff auf Null Pointer

0xC000001D : "ILLEGAL INSTRUCTION"

Es sollte ein Befehl ausgeführt werden, der nicht definiert ist.

0xC000008C : "ARRAY BOUNDS EXCEEDED"

Zugriff außerhalb der definierten Array-Grenzen

0xC000008D : FLOAT DENORMAL OPERAND
 0xC000008E : FLOAT DIVIDE BY ZERO
 0xC000008F : FLOAT INEXACT RESULT
 0xC0000090 : FLOAT INVALID OPERATION
 0xC0000091 : FLOAT OVERFLOW
 0xC0000092 : FLOAT STACK CHECK
 0xC0000093 : FLOAT UNDERFLOW
 0xC0000094 : INTEGER DIVIDE BY ZERO
 0xC0000095 : INTEGER OVERFLOW

Es soll ein Resultat berechnet werden, das aber nicht berechnet werden kann, & weil die Berechnung mit den aktuellen Werten ein ungültiges Ergebnis erzeugt.

0xC0000096 : PRIVILEGED INSTRUCTION

Es sollte ein privilegierter Befehl ausgeführt werden, der nicht erlaubt ist.

0xC00000FD : STACK_OVERFLOW

Ein Stack ist übergelaufen.

Eintrag Real-time RUNTIME Error;Task=WSP0_RT;Object=CALLVARAN1

Beschreibung Gesamtdauer aller Realtime Objekte überschreitet maximale Zeit.

2 ms bei 386er CPUs

1 ms bei restlichen CPUs

Fehler trat im Task WSP0_RT (Realtime Task) auf.

Fehler trat im Object CALLVARAN1 auf.

Eintrag hat auch einen Dump Eintrag => siehe Exception

Ursache / Abhilfe Echtzeit Task der Applikation optimieren (RtWork).

Echtzeit Task Taktzeit aller Objekte verlangsamen.

Applikationsfehler beheben

CPU ist im Realtime zu ausgelastet => Leistungsstärkere CPU verwenden.

Eintrag RUNTIME Error;Task=WSP0_CT;Object=AXLESCHEDULER1

Beschreibung Gesamtdauer aller zyklischer Objekte überschreitet maximale Zeit.

Fehler trat im Task WSP0_CT (Cyclic Task) auf.

Fehler trat im Object AXLESCHEDULER1 auf.

Eintrag hat auch einen Dump Eintrag => siehe Exception

Ursache / Abhilfe	<p>Zyklischen Task der Applikation optimieren (CyWork). Zeit kann durch 2 globale Systemvariablen konfiguriert werden:</p> <ul style="list-style-type: none"> • Runtime: Verbleibende Restzeit • SWRuntime: Vorwahlwert für Runtime-Zähler <p>Oder mittels „autoexec.lsl“ eingestellt werden (SET RUNTIME 30) Leistungsstärkere CPU verwenden</p>
Eintrag Beschreibung	<p>Background RUNTIME Error;Task=WSP0_BT;Object=CLASS01</p> <p>Gesamtdauer aller Background Objekte überschreitet maximale Zeit. Fehler trat im Task WSP0_BT (Background) auf. Fehler trat im Object CLASS01 auf. Eintrag hat auch einen Dump Eintrag => siehe Exception</p>
Ursache / Abhilfe	<p>Background Task der Applikation optimieren (Background) Zeit kann durch 2 globale Systemvariablen konfiguriert werden:</p> <ul style="list-style-type: none"> • BTRuntime: Verbleibende Restzeit • SWBTRuntime: Vorwahlwert für Runtime-Zähler <p>Leistungsstärkere CPU verwenden</p>
Eintrag Beschreibung	<p>*** overflow at 08/05/08;09:51:09.214; ***</p> <p>Zu viele Logeinträge auf einmal. Ein oder mehrere Logeinträge gingen verloren.</p>
Ursache / Abhilfe	<p>Info</p>
Eintrag Beschreibung	<p>F A T A L S Y S T E M E R R O R Signal: Resource released out of sequence Current task : Main Task</p> <p>Fataler System Fehler</p>
Ursache / Abhilfe	<p>Programmierfehler Applikation beheben</p>

8.2.2 USB

Eintrag Beschreibung	<p>USB;2;USB connect VID=0000 PID=0000 USB;2; Class=9 SubClass=0 Protocol=0</p> <p>USB-Gerät wurde erkannt und verbunden mit VendorID 0000 und ProductID 0000 Class=9 SubClass=0 Protocol=0</p>
-------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Ursache / Abhilfe	USB-Gerät gefunden und verbunden Hinweis: Auch Steuerungsinterne Geräte werden hier aufgelistet
Eintrag	USB;2;USB disconnect VID=046d PID=c30e
Beschreibung	USB-Gerät wurde entfernt
Ursache / Abhilfe	USB-Stick oder USB-Tastatur abgesteckt

8.2.3 DIAS

Eintrag	DIASMaster: PlTime = 4
Beschreibung	DIAS-Bus Taktzeit ist 4 ms
Ursache / Abhilfe	Info
Eintrag	DIASMaster: DIASType = 2, CdiasType = 2
Beschreibung	Art des DIAS-Busses: 0 = Kein 1 = Standart Master 2 = Intelligenter Master Art des C-DIAS-Busses: 0 = Kein 1 = Memory mapped 2 = Memory mapped
Ursache / Abhilfe	Info
Eintrag	DIASMaster: PLL is locked and Rt program of intelligent master starts (1)
Beschreibung	Hardware Klasse synchron zur Hardware und startet nun den Intelligenten Master.
Ursache / Abhilfe	Info
Eintrag	DIASMaster: We are Sync -> Rt program of intelligent master starts again (Sync Module Place: 0) (2)
Beschreibung	Hardwareklasse Synchron. Antwort des DIAS-Bus Sync – Befehls wird auf Modulnummer mit Stationsnummer 0 installiert. Bei 'Sync Module Place: 255' wurde am DIAS-Bus kein Modul gefunden welches den DIAS-Bus Sync – Befehl verwendet.
Ursache / Abhilfe	Info
Eintrag	DIASMaster: MoveStartPointer -> Start moment of IM - program is set (Rt - Startpoint: 3900us, AddTime: 0ns) (3)

Beschreibung	Startpunkt des Intelligenten Masters wurde auf 3900 µs gesetzt. Für jedes CIC mit Repeater (CIC012/121/212/221) wird zusätzliche Zeit benötigt, um das Signal aufzubereiten. Diese Zeit wird als AddTime angegeben.
Ursache / Abhilfe	Info
Eintrag	DIASMaster: One Time point was read -> start updating realtime tasks now (0 Rt - DIAS accesses) (4)
Beschreibung	Es wurde die Zeit bis zum nächsten Durchlauf des Intelligenten Masters ausgelesen und ab diesem Zeitpunkt werden die Echtzeit-Hardwareklassen bearbeitet, wobei die Anzahl der DIAS-Zugriffe angegeben wird (in diesem Fall = 0).
Ursache / Abhilfe	Info
Eintrag	DIASMaster: Init ends
Beschreibung	Initialisierung des DIAS-Busses abgeschlossen
Ursache / Abhilfe	Info
Eintrag	DIASError on DIV511 (DeviceAddress: 10000, DIASModulePlace: 9)
Beschreibung	DIAS-Error am DIAS-Bus des DIV511 mit Device Adresse hex 10000 auf Stationsnummer 9 ist aufgetreten. Bei DIAS-Error auf Station 63 ist der DIAS-Bus Sync – Befehl nicht beantwortet worden.
Ursache / Abhilfe	Falls die Stationsnummer im Netzwerk nicht vergeben ist, kann der DIAS Error auch ein Hardwareklassenfehler sein. Kann beim Ausschalten des Systems auftreten. Hardwareproblem Siehe auch Kapitel Fehlersuchhilfe für DIAS-Bus-Probleme
Eintrag	DIASMaster: Unable to create semaphore for async access!
Beschreibung	Es konnte kein Semaphor für die OS-Aufrufe angelegt werden.
Ursache / Abhilfe	Kein weiteres Semaphor mehr möglich oder mehrere DIASMasterC Objekte platziert.
Eintrag	DIASMaster: Wrong DIASMaster Time for new PLL! SystemPeriod: %d, DIASMasterTime: %d
Beschreibung	DIASMasterTime muss ein Vielfaches von (SystemPeriod / 100) sein.
Ursache / Abhilfe	Realtime-Zeit von DIASMaster-Objekt auf ein Vielfaches der Systemperiodenzeit einstellen. (Default: 1ms)
Eintrag	Watchdog Error (RTSSW<>4) -> DIASMasterOn = 0 and lmOff = 1

Beschreibung	DIAS/C-DIAS Watchdog wurde nicht getriggert, weil das DIASMaster-Objekt für mehr als 30 ms nicht aufgerufen wurde. -> Hardware-Reset am DIAS- und C-DIAS Bus. Diese Meldung wird ausgegeben, wenn die Initialisierung des DIASMaster-Objekts noch nicht abgeschlossen ist.
Ursache / Abhilfe	Programmfehler beheben. Eventuell Endlosschleife in einer Init-Methode.
Eintrag	Watchdog Error -> DIASMasterOn = 0 and ImOff = 1
Beschreibung	DIAS/C-DIAS Watchdog wurde nicht getriggert, weil das DIASMaster-Objekt für mehr als 30 ms nicht aufgerufen wurde. -> Hardware-Reset am DIAS und C-DIAS Bus. Diese Meldung wird ausgegeben, wenn die Initialisierung des DIASMaster-Objekts bereits abgeschlossen ist.
Ursache / Abhilfe	Programmfehler beheben. Eventuell Endlosschleife in einem Realtime Task.
Eintrag	DIASMaster: Watchdog on C-DIAS and DIAS - Bus occured! (The RtTask of DIASMasterC - Class was set Off for more than 30ms => Hardwarereset on DIAS and CDIAS - Bus)
Beschreibung	Der Watchdog wurde 130 ms nicht getriggert. Die C-DIAS / DIAS-Bus geht in den Fail Safe Zustand.
Ursache / Abhilfe	Der Realtime Task wurde mindestens 30 ms blockiert. Tritt daher meist in Kombination mit einem Realtime Runtime Fehler auf.
Eintrag	DIASMaster: Timeout on asynchron methods! => DIASBus is set off!!
Beschreibung	Bei den asynchronen DIAS-Bus-Befehlen des Intelligenzen Masters kam es zu einem Timeout.
Ursache / Abhilfe	Intelligerter Master ist ausgeschaltet (ev. aufgrund eines Watchdog Fehlers). Folgefehler.
Eintrag	DIASMaster: But no Watchdog on CDIAS and DIAS-Bus
Beschreibung	Status des Watchdog zum vorherigen Logeintrag.
Ursache / Abhilfe	Info
Eintrag	DIASMaster: RISC is set of -> DIASMasterOn = 0 and ImOff = 1
Beschreibung	Intelligerter Master wurde ausgeschaltet.
Ursache / Abhilfe	Folgefehler / Info
Eintrag	DIASMaster: Rt of IM starts while Rt of DIASMaster is running!

Beschreibung	Realtime Programm des Intelligenten Masters startet während die Hardwareklassen noch laufen.
Ursache / Abhilfe	Zeitliches Problem des Realtimeprogramms der Applikation. RtWork der Applikation optimieren. Leistungsstärkere CPU einsetzen.
Eintrag	DIASMaster: Timeslice overrun!
Beschreibung	Zeitscheibenverletzung im Intelligenten Master
Ursache / Abhilfe	Zeitliches Problem des Realtimeprogramms der Applikation. RtWork der Applikation optimieren. Leistungsstärkere CPU einsetzen.
Eintrag	DIASMaster: Too many cyclic calls
Beschreibung	DPRAM des Intelligenten Masters für Cyclic Objekte ist voll.
Ursache / Abhilfe	Mehr HW-Klassen auf Realtime stellen.
Eintrag	DIASMaster: Too many realtime calls
Beschreibung	DPRAM des Intelligenten Masters für Realtime Objekte ist voll.
Ursache / Abhilfe	Mehr HW-Klassen auf Cyclic stellen.
Eintrag	DIASMaster: Cyclic program of intelligent master is too large
Beschreibung	DPRAM des Intelligenten Masters für Cyclic Objekte ist voll.
Ursache / Abhilfe	Mehr HW-Klassen auf Realtime stellen.
Eintrag	DIASMaster: Realtime program of intelligent master is too large
Beschreibung	DPRAM des Intelligenten Masters für Realtime Objekte ist voll.
Ursache / Abhilfe	Mehr HW-Klassen auf Cyclic stellen.
Eintrag	DIASMaster: Intelligent master is setting Off
Beschreibung	Intelligenter Master wurde aufgrund des vorhergehenden Logeintrags ausgeschaltet.
Ursache / Abhilfe	Info

Eintrag	DIASMaster: Timeout while synchronization
Beschreibung	Problem beim Synchronisieren mit der Hardware.
Ursache / Abhilfe	Realtime Zeit ist ausgeschaltet => auf z.B. 1 ms setzen
Eintrag	DIAS Error (9)
Beschreibung	DIAS-Error auf Stationsnummer 9 (Nummer ist in Hex => eingestellte Nummer der Hexschalter) Bei DIAS-Error auf Statuson 3F ist der DIAS-Bus Sync – Befehl nicht beantwortet worden. Das Modul, welches nicht geantwortet hat, kann aus dem Logeintrag „DIASMaster: We are Sync -> Rt programm of intelligent master starts again (Sync Module Place: 0) (2)“ Sync Modul Place entnommen werden (in diesem Fall 0 , Vorsicht: diese Zahl ist in dezimal).
Ursache / Abhilfe	Falls die Stationsnummer im Netzwerk nicht vergeben ist, kann der DIAS Error auch ein Hardwareklassenfehler sein. Kann beim Ausschalten des Systems auftreten. Hardwareproblem Siehe auch Kapitel Fehlersuchhilfe für DIAS-Bus-Probleme
Eintrag	DIAS Risc Error (6)
Beschreibung	Software Bedienungsfehler des Intelligenten Masters. Die Nummer ist nicht relevant.
Ursache / Abhilfe	SIGMATEK kontaktieren

8.2.4 S-DIAS

Eintrag	SDIAS_LOCAL;2;PCI config interface not found!
Beschreibung	Es ist kein PCI-Interface vorhanden, wodurch auch keine PCI-Geräte unterstützt werden.
Ursache / Abhilfe	Info
Eintrag	SDIAS_LOCAL;2;SDIAS_MANAGER_CFG Space in FPGA not found!
Beschreibung	Im FPGA ist kein S-DIAS-Manager vorhanden.
Ursache / Abhilfe	Es ist kein lokaler S-DIAS-Bus vorhanden. Die S-DIAS-Clients können nur hinter Bus-Koppelmodulen verwendet werden (wie z.B. VI021).
Eintrag	SDIAS_LOCAL;2;VARAN_PLL_REGISTER Space in FPGA not found!
Beschreibung	Die Nanosekunden-PLL, welche für den lokalen S-DIAS-Manager benötigt wird, ist nicht vorhanden.

Ursache / Abhilfe	Der lokale S-DIAS-Manager kann nicht verwendet werden, da das FPGA nicht konsistent ist.
Eintrag	SDIAS_LOCAL;2;Found ctrl=%p, dpram=%p, pll=%p
Beschreibung	Auflistung der virtuellen Startadressen für die Komponenten im FPGA, die für den S-DIAS-Manager benötigt werden.
Ursache / Abhilfe	Info
Eintrag	SDIAS_LOCAL;2:length bigger than area
Beschreibung	Beim Kopieren der isochronen Daten vom Shadow-Buffer in das DPRAM bzw. umgekehrt, ist ein Fehler aufgetreten, da die Länge der Quelldaten die Größe des Zielbuffers übersteigt.
Ursache / Abhilfe	<p>Die Länge der Daten der Datenobjekte überschreitet die Größe des zur Verfügung stehenden DPRAMs bzw. des Shadow-Buffers.</p> <p>Die Anzahl der Bytes in den isochronen Datenobjekten ist zu reduzieren. Dies kann entweder durch die Änderung der Topologie (z.B. Entfernung von nicht benötigten S-DIAS-Modulen, Platzierung von S-DIAS-Modulen vom lokalen S-DIAS-Bus am S-DIAS-Bus hinter einem Bus-Koppelmodul) oder durch Optimierung der S-DIAS-Speicherzugriffe erfolgen.</p>
Eintrag	SDIAS_LOCAL;3;No SDIAS bus available
Beschreibung	Es steht kein S-DIAS-Manager für den lokalen S-DIAS-Bus zur Verfügung.
Ursache / Abhilfe	<p>Es konnte kein S-DIAS-Manager für den lokalen S-DIAS-Bus initialisiert werden, da entweder nicht alle hierfür benötigten Hardware-Komponenten vom System erkannt wurden oder bei der Initialisierung dieser Fehler aufgetreten sind.</p> <p>Dieser Log-Eintrag deutet daraufhin, dass die CPU keinen lokalen S-DIAS-Bus unterstützt oder, sollte das nicht der Fall sein, ein Hardwaredefekt vorliegt.</p>
Eintrag	SDIAS_LOCAL;2;SDIAS Manager task creation failed!
Beschreibung	Der Task für die Datenverarbeitung im lokalen S-DIAS-Manager konnte nicht erzeugt werden.
Ursache / Abhilfe	<p>Aufgrund fehlender Ressourcen konnte der Task für den S-DIAS-Manager nicht angelegt und gestartet werden.</p> <p>Die Größe des OS-Heaps ist zu gering gewählt und muss entsprechend vergrößert werden.</p>
Eintrag	SDIAS_LOCAL;2;SDIAS subsystem running...
Beschreibung	Der S-DIAS-Manager für den lokalen S-DIAS-Bus konnte erfolgreich initialisiert und die entsprechende Verarbeitung gestartet werden.
Ursache / Abhilfe	Info

Eintrag	SDIAS_LOCAL;2;SDO request failed with global error: channel %d, client: %u, command: 0x%04X, error code: 0x%08X
Beschreibung	Der Zugriff auf einen S-DIAS-Client ist fehlgeschlagen.
Ursache / Abhilfe	<p>Für die Analyse der genaueren Ursache für den fehlgeschlagenen Zugriff werden im Log-Eintrag folgende Informationen angeführt:</p> <ul style="list-style-type: none"> Nummer des Kommunikationskanals (SDO-Kanal) Steckplatz des betreffenden S-DIAS-Clients Fehlgeschlagenes Kommandos Fehler-Code
Eintrag	SDIAS_LOCAL;2;Mutex creation failed
Beschreibung	Ein für die SDO-Kommunikation mit S-DIAS-Teilnehmern benötigtes Mutex konnte nicht erstellt werden.
Ursache / Abhilfe	<p>Aufgrund fehlender Ressourcen konnte das Mutex nicht angelegt bzw. erstellt werden.</p> <p>Die Größe des OS-Heaps ist zu gering gewählt und muss entsprechend vergrößert werden.</p>
Eintrag	SDIAS_LOCAL;1;SDIAS_Manager_API_Init: SDIAS SDO task create failed
Beschreibung	Der Task für die SDO-Kommunikation mit S-DIAS-Teilnehmern konnte nicht erstellt werden.
Ursache / Abhilfe	<p>Aufgrund fehlender Ressourcen konnte der Task für die SDO-Kommunikation nicht angelegt und gestartet werden.</p> <p>Die Größe des OS-Heaps ist zu gering gewählt und muss entsprechend vergrößert werden.</p>
Eintrag	SDIAS_LOCAL;2;SDIAS_Manager_Task_Execute: error
Beschreibung	Bei der Ausführung einer asynchronen Task-Liste im S-DIAS-Manager ist ein Fehler aufgetreten.
Ursache / Abhilfe	Info
Eintrag	SDIAS_LOCAL;2;SDIAS_Module_CPLD_Com_Write_Data: request outside of SDIAS manager control detected (%lu)
Beschreibung	Ein Schreibzugriff auf den Speicher des S-DIAS-Managers ist fehlgeschlagen.
Ursache / Abhilfe	Es wurde versucht, auf eine ungültige Adresse im S-DIAS-Manager zuzugreifen. Für die Erkennung des fehlerhaften Zugriffs wird die Startadresse des Schreibzugriffs im Log-Eintrag mitausgegeben.
Eintrag	SDIAS_LOCAL;2;SDIAS_Module_CPLD_Com_Read_Data: request outside of SDIAS manager control detected (%lu)
Beschreibung	Ein Lesezugriff auf den Speicher des S-DIAS-Managers ist fehlgeschlagen.

Ursache / Abhilfe	Es wurde versucht, auf eine ungültige Adresse im S-DIAS-Manager zuzugreifen. Für die Erkennung des fehlerhaften Zugriffs wird die Startadresse des Lesezugriffs im Log-Eintrag mitausgegeben.
Eintrag	SDIAS_LOCAL;2;SDIASManager: iHWTAddModule2Node failed with %d for device %d
Beschreibung	Ein S-DIAS-Client konnte nicht zum Hardwaretree hinzugefügt werden.
Ursache / Abhilfe	Die Ursache hierfür ist als Returncode in der Log-Meldung enthalten, ebenso wie der Steckplatz des betreffenden S-DIAS-Clients.
Eintrag	SDIAS_LOCAL;2;-----> node %p, rootnode %p, pVNode %p, pSdiasManager %p
Beschreibung	Es wurde ein S-DIAS-Client im Hardwaretree hinzugefügt. Der Log-Eintrag beinhaltet die Adresse des eingefügten und die Adressen des darüberliegenden Modulknotens.
Ursache / Abhilfe	Info
Eintrag	SDIAS_LOCAL;2;SAFETY_LVDS: Add module (0x%x/0x%x) to the SafetyCPU BUSINTERNAL tree
Beschreibung	Es wurde eine Erweiterungskarte der S-DIAS-Safety-CPU in den Hardwaretree für den Bus-Typ 0x92 eingefügt. Die Device- und Sub-Device-ID der eingefügten Erweiterungskarte ist im Log-Eintrag enthalten.
Ursache / Abhilfe	Info
Eintrag	SDIAS_LOCAL;2;SAFETY_LVDS: Add module (0x%x/0x%x) to the SafetyCPU CANBUS tree
Beschreibung	Es wurde ein CAN-Bus-Teilnehmer der S-DIAS-Safety-CPU in den Hardwaretree für den Bus-Typ 0x87 eingefügt. Die Device- und Sub-Device-ID des eingefügten CAN-Bus-Teilnehmers ist im Log-Eintrag enthalten.
Ursache / Abhilfe	Info
Eintrag	SDIAS_LOCAL;2;SAFETY_LVDS: Found a module (0x%x/0x%x) with unknown bus topology! Node will be ignored.
Beschreibung	Es wurde ein Erweiterungsmodul der S-DIAS-Safety-CPU mit einem nicht unterstützten Bus-Typ erkannt. Die Device- und Sub-Device-ID des Erweiterungsmoduls ist im Log-Eintrag enthalten.
Ursache / Abhilfe	Der ermittelte Bus-Typ wird vom Betriebssystem nicht erkannt. Dies kann z.B. daran liegen, dass die eingesetzte OS-Version den gefundenen Bus-Typ noch nicht unterstützt (in diesem Fall muss das Betriebssystem aktualisiert werden) oder ein Hardwaredefekt der Safety-CPU vorliegt.
Eintrag	SDIAS_LOCAL;2;SDIAS_Module_Obj_Config_Memory_Get failed for SAFETY CPU

Beschreibung	Das Auslesen der Kennungen der Erweiterungskarten einer S-DIAS-Safety-CPU ist fehlgeschlagen.
Ursache / Abhilfe	Beim Speicherzugriff auf die S-DIAS-Safety-CPU ist ein Fehler aufgetreten. Als mögliche Ursachen kommen hier z.B. defekte Hardware, schlechte Steckkontakte oder Störungen in Frage.
Eintrag	(BusInterfaceVARAN::Init) GetLongDOsPossible was called with the wrong VARANManagerNr.
Beschreibung	Am VARAN-Manager wurde eine VARAN-Manager-Nummer ungleich 0 angegeben. Die Verfügbarkeit von LongDOs kann nicht ermittelt werden.
Ursache / Abhilfe	Info
Eintrag	(BusInterfaceVARAN::AddDO) The FPGA of the VARAN Manager does not support Long DOs. Newer Version required!
Beschreibung	Der VARAN-Manager unterstützt noch keine „LongDOs“ (Datenobjekte > 128 Byte).
Ursache / Abhilfe	Neuere FPGA-Version für die entsprechende Plattform verwenden.
Eintrag	(BusInterfaceVARAN::BusInterfaceVARAN) Failed to get VARAN-CIL (interface to OS)!
Beschreibung	Betriebssysteminterface für VARAN konnte nicht gefunden werden.
Ursache / Abhilfe	Neuere OS-Version verwenden.
Eintrag	(BusInterfaceSDIASInternal::SetSyncData) CycleTime of SDIAS Manager has to be a multiple of the maintimer setting!
Beschreibung	Die eingestellte S-DIAS-Buszeit ist kein Vielfaches des CPU Maintimers.
Ursache / Abhilfe	Maintimer oder S-DIAS-Buszeit anpassen.
Eintrag	(BusInterfaceSDIASInternal::SetSyncData) HwControl realtime setting has to be a multiple of the maintimer setting!
Beschreibung	Die eingestellte Realtime Zeit ist kein Vielfaches des CPU Maintimers.
Ursache / Abhilfe	Maintimer oder Realtime Zeit anpassen.
Eintrag	(BusInterfaceSDIASInternal::SetSyncData) Invalid constellation of MainTimer and cycle time of SDIAS! CycleTime/MainTimer has to be smaller than 256!
Beschreibung	Die eingestellte S-DIAS-Buszeit ist zu groß für den gewählten Maintimer der CPU.

Ursache / Abhilfe	S-DIAS-Buszeit [μs] / Maintimer [μs] muss kleiner als 256 sein.
Eintrag	(SdiasManager::SdiasManager) Failed to get SDIAS OS-Interface. A newer OS is necessary to use SDIAS
Beschreibung	Das Betriebssysteminterface für S-DIAS konnte nicht geholt werden.
Ursache / Abhilfe	Die verwendete Betriebssystemversion unterstützt noch keinen S-DIAS-Bus. Neuere Betriebssystemversion verwenden.
Eintrag	(BusInterfaceSDIASInternal::HwControlLogin) CycleTime of SDIAS Manager has to be a multiple of the HwControl realtime setting!
Beschreibung	Die eingestellte S-DIAS-Buszeit ist kein Vielfaches der Realtimezeit des HWControl Objekts.
Ursache / Abhilfe	S-DIAS-Buszeit oder Realtime Zeit anpassen.
Eintrag	(BusInterfaceSDIASInternal::HwControlLogin) Response of login at HwControl has an invalid size
Beschreibung	Die Version der Klasse ist nicht mit der verwendeten HWControl kompatibel.
Ursache / Abhilfe	Hardwareklassen updaten.
Eintrag	(BusInterfaceSDIASInternal::HwControlLogin) Failed to install callbacks at HwControl
Beschreibung	Die verwendete Version der HWControl ist zu alt.
Ursache / Abhilfe	HWControl updaten.
Eintrag	(SdiasManager::Init) SDIAS Manager object inactive";
Beschreibung	Der S-DIAS-Manager läuft nicht bzw. ist deaktiviert.
Ursache / Abhilfe	Bei lokalem S-DIAS-Bus: Folgefehler, wenn das Betriebssysteminterface für S-DIAS nicht gefunden wurde. Betriebssystem updaten. S-DIAS-Bus über VARAN (z.B.: VI021) Entweder der VARAN-Manager ist aufgrund eines Fehlers abgeschalten oder das VI021 ist auf Transparent gestellt.
Eintrag	(SdiasManager::SdiasManager) Failed to get SDIAS OS-Interface. A newer OS is necessary to use SDIAS";
Beschreibung	Das Betriebssysteminterface für S-DIAS konnte nicht geholt werden.
Ursache / Abhilfe	Die verwendete Betriebssystemversion unterstützt noch keinen S-DIAS-Bus.

	Neuere Betriebssystemversion verwenden.
Eintrag	(SdiasManager::Init) Failed to get bus cycle time via BusInterface");
Beschreibung	Die Buszykluszeit konnte nicht ermittelt werden.
Ursache / Abhilfe	Folgefehler, wenn die eingestellten Bus- und Realtime-Zeiten nicht möglich sind. Siehe weitere Logeinträge für nähere Informationen.
Eintrag	(SdiasManager::Init) Failed to add DO for the asynchronous read data");
Beschreibung	Ein Datenobjekt konnte nicht erstellt werden.
Ursache / Abhilfe	Siehe weitere Logeinträge für nähere Informationen.
Eintrag	(SdiasManager::Init) Failed to add DO for the asynchronous write data");
Beschreibung	Ein Datenobjekt konnte nicht erstellt werden.
Ursache / Abhilfe	Siehe weitere Logeinträge für nähere Informationen.
Eintrag	(SdiasManager::CyWork) No SDIAS Client objects projected/connected to the SDIAS Manager! Unable to continue with Initialisation!");
Beschreibung	Es wurden keine S-DIAS-Client Objekte in der Software gefunden/angeschlossen.
Ursache / Abhilfe	Dient nur zur Information. Die Initialisierung des S-DIAS-Busses wird nicht fortgesetzt, die restliche Hardware ist nicht davon betroffen.
Eintrag	(SdiasManager::RtWork) Error at toggle bit of iso write task of SDIAS");
Beschreibung	Das Togglebit des isochronen Schreibtasks hat den falschen Zustand.
Ursache / Abhilfe	Der Schreibtask wurde nicht fertig ausgeführt. Einstellung der Buszeit und des ISOStartPoints überprüfen.
Eintrag	(SdiasManager::RtWork) Error at toggle bit of iso read task of SDIAS");
Beschreibung	Das Togglebit des isochronen Lesetasks hat den falschen Zustand.
Ursache / Abhilfe	Der Lesetask wurde nicht fertig ausgeführt. Einstellung der Buszeit und des ISOStartPoints überprüfen.
Eintrag	(SdiasManager::RtWork) Error at iso write task of SDIAS");
Beschreibung	Der isochrone Schreibtask hat einen Fehler festgestellt.

Ursache / Abhilfe	Bei einem Schreibzugriff ist ein Fehler aufgetreten. <ul style="list-style-type: none"> • Busverbindung schlecht / gestört • Versorgungsspannung ausgefallen oder Einbrüche
Eintrag	(SdiasManager::RtWork) Iso write task of SDIAS hasn't been completed");
Beschreibung	Der isochrone Schreitask wurde im letzten Zyklus nicht fertig.
Ursache / Abhilfe	Nicht genügend Zeit für den Schreitask. Einstellung der Buszeit und des ISOStartPoints überprüfen.
Eintrag	(SdiasManager::RtWork) Iso read task of SDIAS hasn't been completed! Check setting of IsoStartPoint and cycle time!");
Beschreibung	Der isochrone Lesetask wurde im letzten Zyklus nicht fertig.
Ursache / Abhilfe	Nicht genügend Zeit für den Lesetask. Einstellung der Buszeit und des ISOStartPoints überprüfen.
Eintrag	(SdiasManager::Init) Missing SDIAS Place 0x{0} in Configuration!
Beschreibung	Ein Hardwaremodul wurde gefunden, für das keine Hardwareklasse platziert ist.
Ursache / Abhilfe	Information: Hardware Steckplatz [hex] auf dem ein Modul gefunden wurde, aber keine Hardwareklasse vorhanden ist.
Eintrag	(SdiasManager::CyWork) Wrong device ID at SDIAS Place 0x{0}", i);
Beschreibung	Das Hardwaremodul stimmt nicht mit der Software überein.
Ursache / Abhilfe	Information: Hardware Steckplatz [hex] der nicht mit der Software übereinstimmt.
Eintrag	(SdiasManager::CyWork) Failed to initialize module on SDIAS Place 0x{0} via InitModule-Interface", ActModule[i]);
Beschreibung	Ein S-DIAS-Modul lieferte einen Fehler bei der Initialisierung.
Ursache / Abhilfe	Information: Siehe weitere Logeinträge für nähere Informationen.
Eintrag	(SdiasManager::CyWork) Init has been blocked for too long! Initialization of SDIAS Manager timed out! Blocked step: 0x{0}",InitSSW\$UDINT);
Beschreibung	Ein S-DIAS-Modul lieferte einen Fehler bei der Initialisierung.
Ursache / Abhilfe	Information: Siehe weitere Logeinträge für nähere Informationen.

Eintrag	(SdiasManager::AddAccess) Invalid Place number: 0x{0}. Maximum is 0x{1}", Place, SDIAS_MAX_PLACE_NR);
Beschreibung	Es wurde eine ungültige S-DIAS-Place Nummer vergeben.
Ursache / Abhilfe	Place-Einstellungen in Software überprüfen.
Eintrag	(SdiasManager::SetRequiredError) Required Error on SDIAS Place 0x{0}", Place);
Beschreibung	Es wurde ein Required Error für ein fehlendes Modul ausgelöst.
Ursache / Abhilfe	<ul style="list-style-type: none"> Hardwareaufbau prüfen, ob alle Module angeschlossen und versorgt sind. Busverbindung prüfen. Versorgungsspannung prüfen

8.2.5 VARAN

Eintrag	MAC-Address not available, VARAN to Ethernet Bridge (0)!								
Beschreibung	Für den VARAN Manager ist keine MAC-Adresse definiert. Es wird die MAC-Adresse der Netzwerkkarte verwendet.								
Ursache / Abhilfe	Info								
Eintrag	Task VARAN Task is terminating!								
Beschreibung	Die Task (hier „VARAN Task“) wurde beendet, weil sie eine Exception ausgelöst hat.								
Ursache / Abhilfe	Fehler im Betriebssystem oder Callback-Funktionen der Hardwareklassen. => Betriebssystem und Hardwareklassen updaten								
Eintrag	VARAN Callback (DEA30)(5) ODER VARAN Callback (TIME SLICE ERROR IRQ: VMC052,ADDR:0x0,Port:0)								
Beschreibung	Der Applikation wurde eine Statusänderung am VARAN-Bus mitgeteilt. Hinterste Zahl in der alten Variante ist der Statuscode: <table border="1" style="margin-left: 20px;"> <tr> <td>0</td> <td>Device wurde verbunden (CONNECT oder CONNECT PREV)</td> </tr> <tr> <td>1</td> <td>Device wurde entfernt (DISCONNECT)</td> </tr> <tr> <td>2</td> <td>Zugriffsfehler auf ein required Modul (required-frame error)</td> </tr> <tr> <td>3</td> <td>Zugriffsfehler auf ein not required Modul (not-required-frame error)</td> </tr> </table>	0	Device wurde verbunden (CONNECT oder CONNECT PREV)	1	Device wurde entfernt (DISCONNECT)	2	Zugriffsfehler auf ein required Modul (required-frame error)	3	Zugriffsfehler auf ein not required Modul (not-required-frame error)
0	Device wurde verbunden (CONNECT oder CONNECT PREV)								
1	Device wurde entfernt (DISCONNECT)								
2	Zugriffsfehler auf ein required Modul (required-frame error)								
3	Zugriffsfehler auf ein not required Modul (not-required-frame error)								

	<p>4 Fataler System Fehler (FATAL ERROR) 5 Zeitscheibenverletzung der Applikation (TIME SLICE ERROR IRQ) 6 Watchdog (WATCHDOG ERROR)</p> <p>In der neuen Variante ist der Statuscode schon in Klartext + Device auf dem der Fehler aufgetreten ist.</p> <p>Ursache / Abhilfe</p> <p>Statuscode.</p> <p>0.1 Info</p> <p>2.3 Hardwareprobleme: Siehe auch Kapitel Fehlersuchhilfe für VARAN-Bus-Probleme Kann beim Ausschalten des Systems auftreten oder Hardwareprobleme: siehe Kapitel Fehlersuchhilfe für VARAN-Bus-Probleme.</p> <p>5.6 Zeitliches Problem des Realtimeprogramms der Applikation. RtWork der Applikation optimieren. Leistungsstärkere CPU einsetzen. Langsamere Taktzeit aller RtWork einstellen. Kann beim Ausschalten des Systems auftreten. Kann beim Beenden der Applikation (Reset, Runtime, Division by 0,) auftreten.</p>
<p>Eintrag</p> <p>Beschreibung</p> <p>Ursache / Abhilfe</p>	<p>VARAN Watchdog Error</p> <p>Der Watchdog wurde 130 ms nicht getriggert. Der VARAN-Bus geht in den Fail Safe Zustand.</p> <p>Der Realtime Task wurde mindestens 30 ms blockiert. Tritt daher meist in Kombination mit einem Realtime Runtime Fehler auf. Zeitliches Problem des Realtimeprogramms der Applikation. RtWork der Applikation optimieren. Leistungsstärkere CPU einsetzen.</p>
<p>Eintrag</p> <p>Beschreibung</p> <p>Ursache / Abhilfe</p>	<p>VARANManager 0: Started with Version 1.16</p> <p>VARAN Manager Klasse startete mit Version 1.16</p> <p>Info</p>
<p>Eintrag</p> <p>Beschreibung</p> <p>Ursache / Abhilfe</p>	<p>VARANManager 0: VARANManagerTime (%d) is not a multiple of MainTimer (%d)!</p> <p>Die VARAN-Buszykluszeit muss ein Vielfaches des Maintimers (Taktgeber des Betriebssystems und somit schnellstmögliche Zykluszeit) sein.</p> <p>VARAN-Buszykluszeit auf ein Vielfaches des OS-Maintimers setzen oder Maintimer entsprechend ändern. Default für Maintimer und VARAN-Zykluszeit ist 1 ms.</p>

Eintrag	VARANManager 0: Client - Function not available but activated! (at least OS - Version 1.1.217 required)
Beschreibung	VARAN Manager Client Funktion eingeschaltet aber mit dieser Betriebssystem Version nicht verfügbar.
Ursache / Abhilfe	Betriebssystem auf 1.1.217 oder größer updaten
Eintrag	VARANManager 0: VARANManagerTime (%d) is not a multiple of System Period Time (%d)!
Beschreibung	Die VARAN-Buszykluszeit muss ein Vielfaches der Systemperiode (= Taktzeit des VARANManager-FPGAs) sein.
Ursache / Abhilfe	VARAN-Buszykluszeit auf ein Vielfaches der FPGA-Systemperiode setzen.
Eintrag	VARANManager 0: Unable to create semaphore for async access!
Beschreibung	Es konnte kein Semaphor für Direct Accesses angelegt werden.
Ursache / Abhilfe	Kein weiteres Semaphor mehr möglich oder mehrere VARANManager Objekte platziert.
Eintrag	VARANManager 0: IRQ Task Time not a multiple of VARANManager Time!
Beschreibung	VARAN Manager Zeit muss ein Vielfaches der Interrupt Task Zeit sein!
Ursache / Abhilfe	Korrigieren der IRQ Task Time.
Eintrag	VARANManager 0: No Memory for VARAN Manager available! (Make Rt - Handles: 50:
Beschreibung	Es konnte kein Speicher allokiert werden für die Realtime-Handles (damit werden alle UpdateRt-Methoden der VARAN-Hardwareklassen aufgerufen).
Ursache / Abhilfe	Weniger Speicher in der Applikation reservieren.
Eintrag	VARANManager 0: No Memory for VARAN Manager available! (Make Cy - Handles: 50:
Beschreibung	Es konnte kein Speicher allokiert werden für die Cyclic-Handles (damit werden alle UpdateCy-Methoden der VARAN-Hardwareklassen aufgerufen).
Ursache / Abhilfe	Weniger Speicher in der Applikation reservieren.
Eintrag	VARANManager 0: No Memory for VARAN Manager available! (Extend Rt - Handles: 50:
Beschreibung	Der Speicher für die Realtime-Handles (damit werden alle UpdateRt-Methoden der VARAN-Hardwareklassen aufgerufen) konnte nicht erweitert werden.

Ursache / Abhilfe	Weniger Speicher in der Applikation reservieren.
Eintrag	VARANManager 0: No Memory for VARAN Manager available! (Extend Cy - Handles: 50;)
Beschreibung	Der Speicher für die Cyclic-Handles (damit werden alle UpdateCy-Methoden der VARAN-Hardwareklassen aufgerufen) konnte nicht erweitert werden.
Ursache / Abhilfe	Weniger Speicher in der Applikation reservieren.
Eintrag	VARANManager 0: Error happened with ErrorCode %+d!
Beschreibung	<p>Error has occurred in the VARANManager ErrorCode:</p> <ul style="list-style-type: none"> -1 VARANMANAGER_DRIVER_NOT_EXISTS -2 VARANMANAGER_DOL_TYPE_WRONG -3 VARANMANAGER_RUN_STATUS_WRONG -4 VARANMANAGER_DO_HANDLE_INVALID -5 VARANMANAGER_DO_RAM_FULL -6 VARANMANAGER_DO_CMD_INVALID -7 VARANMANAGER_MANAGER_NOT_EXISTS -8 VARANMANAGER_DOL_ADDRESS_INVALID -9 VARANMANAGER_UNKNOWN_COMMAND -10 VARANMANAGER_COMPONENT_NOT_EXISTS -11 VARANMANAGER_CLIENT_NOT_EXISTS -12 VARANMANAGER_CDIAS_EEPROM_NOT_EXISTS -13 VARANMANAGER_CDIAS_EEPROM_NO_GRANT -14 VARANMANAGER_CDIAS_EEPROM_NACK -15 VARANMANAGER_PORT_NOT_EXISTS -16 VARANMANAGER_PORT_IS_UPLINK -17 VARANMANAGER_PORT_NO_LINK -18 VARANMANAGER_NO_MUTEX -19 VARANMANAGER_NO_TASK -20 VARANMANAGER_ID_NOT_FOUND

	-21 VARANMANAGER_ID_NOT_INITIALIZED -22 VARANMANAGER_INVALID_DEVICE_ADDRESS -23 VARANMANAGER_CALLBACK_NOT_HANDLED -24 VARANMANAGER_NO_MEM -25 VARANMANAGER_NO_LEGACY_WD -26 VARANMANAGER_ADMIN_DOL_EXECUTION_ERROR -27 VARANMANAGER_DA_DOL_EXECUTION_ERROR -28 VARANMANAGER_SPI_FLASH_NO_ACCESS -29 VARANMANAGER_CLIENT_NOT_READY -30 VARANMANAGER_CLIENT_DISABLED -31 VARANMANAGER_CLIENT_CANT_ENABLE
Ursache / Abhilfe	Info
Eintrag	VARANManager 0: Is set off because of last Error!
Beschreibung	Manager wurde aufgrund des letzten Fehlers gestoppt.
Ursache / Abhilfe	Info
Eintrag	VARANManager 0: Timeslice Error! Manager is set off!
Beschreibung	Echzeitprogramm des VARAN Managers ist noch nicht fertig wenn die CPU ihren Echtzeittask startet.
Ursache / Abhilfe	ISOStartPoint Einstellung am VARAN Manager richtigstellen. Langsameren Takt einstellen.
Eintrag	VARANManager 0: Timeslice Error Interrupt, Manager is set off!
Beschreibung	Der Echtzeittask der Hardwareklassen ist noch nicht mit dem Bearbeiten der Daten im DPRAM fertig wenn der VARAN Manager seinen Echtzeittask starten will.
Ursache / Abhilfe	ISOStartPoint Einstellung am VARAN Manager richtigstellen. Langsameren Takt einstellen. RtWork der Applikation optimieren.
Eintrag	VARANManager 0: Watchdog Error Interrupt, Manager is set off!

Beschreibung	Der Watchdog wurde 130 ms nicht getriggert. Der VARAN-Bus geht in den Fail Safe Zustand.
Ursache / Abhilfe	<p>Der Realtime Task wurde mindestens 30 ms blockiert. Tritt daher meist in Kombination mit einem Realtime Runtime Fehler auf.</p> <p>Zeitliches Problem des Realtimeprogramms der Applikation.</p> <p>RtWork der Applikation optimieren.</p> <p>Leistungsstärkere CPU einsetzen.</p>
Eintrag	VARANManager 0: Required Module not ready, Manager is set off!
Beschreibung	Ein VARAN-Client-Modul, welches in der Applikation als "required" (= zwingend benötigt) markiert ist, wurde nicht gefunden.
Ursache / Abhilfe	Eventuell Verdrahtungsfehler an der Versorgung oder des VARAN-Buskabels zum entsprechenden Client.
Eintrag	<p>VARANManager 0: Device message</p> <p>DIASError on DIV511 (DeviceAddress: 10000, DIASModulePlace: 9:</p> <p>VARANManager 0: Error on Device, Manager is set off!!</p>
Beschreibung	<p>DIAS-Error am DIAS-Bus des DIV511 mit Device Adresse hex 10000 auf Stationsnummer 9 ist aufgetreten. Bei DIAS-Error auf Statuson 63 ist der DIAS-Bus Sync – Befehl nicht beantwortet worden.</p> <p>DIAS- und VARAN-Bus wechseln in den Fail Safe/Reset Zustand.</p>
Ursache / Abhilfe	<p>Falls die Stationsnummer im Netzwerk nicht vergeben ist, kann der DIAS Error auch ein Hardwareklassenfehler sein.</p> <p>Kann beim Ausschalten des Systems auftreten.</p> <p>Hardwareproblem: Siehe auch Kapitel Fehlersuchhilfe für DIAS-Bus-Probleme</p>
Eintrag	VARANManager 0: SuperiorSystemTime is 0!
Beschreibung	Wenn eine VMC-Einschubkarte verwendet wird, muss am Client SuperiorSystemTime des VARANManagers die Systemzeit des übergeordneten Systems angegeben werden, damit sich die beiden Systeme synchronisieren können.
Ursache / Abhilfe	Zeit des übergeordneten Systems beim Client SuperiorSystemTime des VARANManagers einstellen.
Eintrag	VARANManager 0: Couldn't find sync out for switching alternating buffer
Beschreibung	Für Verwendung von VMC: Die FPGA-Komponente zum Anstoßen des Wechselpufferumschaltens konnte nicht gefunden werden.
Ursache / Abhilfe	Entweder fehlerhafte Hardware oder das Betriebssystem findet fälschlicherweise eine VMC-Schnittstelle.
Eintrag	VARANManager 0: VARANTime of superior system and inferior system have to be multiples of each other

Beschreibung	Für Verwendung von VMC: Die VARAN-Zykluszeit des übergeordneten Systems muss ein Vielfaches der VARAN-Zykluszeit des untergeordneten Systems sein oder umgekehrt, sonst kann keine Synchronisation erfolgen.
Ursache / Abhilfe	Anpassen der Zykluszeit des übergeordneten oder des untergeordneten Systems.
Eintrag	VARANManager 0: VARANTime of superior system and maintimer of inferior system have to be multiples of each other.
Beschreibung	Für Verwendung von VMC: Die VARAN-Zykluszeit des übergeordneten Systems muss ein Vielfaches des OS-Taktes (Maintimer) des untergeordneten Systems sein oder umgekehrt, sonst kann keine Synchronisation erfolgen.
Ursache / Abhilfe	VARAN-Zykluszeit des übergeordneten Systems oder Maintimer des untergeordneten Systems entsprechend anpassen.
Eintrag	VARAN required-frame error ECE06800 ODER VARAN required-frame error DO:ECE06800
Beschreibung	Zugriffsfehler auf ein benötigtes Modul ist aufgetreten. Die Zahl zeigt den Pointer auf das Daten-Objekt mit dem Fehler. Dieser Eintrag ist immer in Kombination mit anderen Einträgen mit welchen das fehlerhafte Modul gefunden werden kann.
Ursache / Abhilfe	Kann beim Ausschalten des Systems auftreten. Hardwareprobleme: Siehe auch Kapitel Fehlersuchhilfe für VARAN-Bus-Probleme
Eintrag	VARAN not-required-frame error ODER VARAN not-required-frame error DO:E80000BC
Beschreibung	Zugriffsfehler auf ein Not Required Modul ist aufgetreten.
Ursache / Abhilfe	Kann beim Ausschalten des Systems auftreten. Kann beim Ausschalten oder Abstecken eines Not Required Moduls auftreten. Hardwareprobleme: Siehe auch Kapitel Fehlersuchhilfe für VARAN-Bus-Probleme
Eintrag	VARAN fatal error DO:E8000724
Beschreibung	Softwareseitiger Bedienungsfehler des VARAN Managers oder Kollision von Datenpaketen am VARAN-Bus.
Ursache / Abhilfe	Kann beim Ausschalten des Systems auftreten. Hardwareprobleme: Siehe auch Kapitel Fehlersuchhilfe für VARAN-Bus-Probleme

Eintrag	VARAN time-slice Error Interrupt				
Beschreibung	<p>Der Echtzeittask der Hardwareklassen ist noch nicht mit dem Bearbeiten der Daten im DPRAM fertig wenn der VARAN Manager seinen Echtzeittask starten will.</p> <p>Dieser Eintrag ist nur gültig, wenn danach der Eintrag „runStatus changed from 0 to 44“ oder „runStatus changed from RUN RAM(0) to VARAN MANAGER ERROR (44)“ und ein Dump (wie bei Exception Error), eingetragen ist.</p>				
Ursache / Abhilfe	<p>Kann beim Ausschalten des Systems auftreten.</p> <p>Kann beim Beenden der Applikation (Reset, Runtime, Division by 0,) auftreten.</p> <p>Zeitliches Problem des Realtimeprogramms der Applikation (RtWork).</p> <p>RtWork der Applikation optimieren.</p> <p>Alle RtWork der Applikation auf eine langsamere Taktzeit stellen.</p> <p>Leistungsstärkere CPU einsetzen.</p>				
Eintrag	VARAN;1;timeout while waiting for admin task				
Beschreibung	Administration Task des Managers ist nach einem Timeout nicht fertig geworden.				
Ursache / Abhilfe	<p>Power On / Off und Meldung an SIGMATEK</p> <p>Kann einmalig in Kombination mit einem “VARAN fatal error“ auftreten.</p>				
Eintrag	<p>VARAN Error L(3),P:010200</p> <p>ODER</p> <p>VARAN Error L(3),P:01-02-00</p>				
Beschreibung	<p>Auf einem VARAN-Modul trat ein Zugriffsfehler auf.</p> <p>Modul ist angesteckt in 3. Ebene (L(3)) => 2 Module zwischen CPU und fehlerhaftem Modul.</p> <p>Modul ist angesteckt am Manager auf Port 1, am folgenden Device auf Port 2, am folgenden Device auf Port 0. Hinweis: Port 0 im Logfile entspricht in Hardware VARAN 1, Port 1 im Logfile entspricht in Hardware VARAN 2,...</p>				
Ursache / Abhilfe	Hardwareprobleme: Siehe auch Kapitel Fehlersuchhilfe für VARAN-Bus-Probleme				
Eintrag	<p>VARAN,EPN:SN01844148</p> <p>ODER</p> <p>VARAN,EAN:SN01996774</p>				
Beschreibung	<p>Auf dem VARAN-Modul mit der Seriennummer 01844148 trat ein Zugriffsfehler auf.</p> <p>Immer in Verbindung mit „VARAN required-frame error“ oder „VARAN not-required-frame error“</p> <table border="0"> <tr> <td style="background-color: #e0e0ff;">EPN</td> <td>die Adresse des Knotens ist noch nicht gesetzt (weil er nicht verbunden ist).</td> </tr> <tr> <td style="background-color: #e0e0ff;">EAN</td> <td>der Knoten hat eine gültige Adresse (Knoten war bis jetzt verbunden).</td> </tr> </table>	EPN	die Adresse des Knotens ist noch nicht gesetzt (weil er nicht verbunden ist).	EAN	der Knoten hat eine gültige Adresse (Knoten war bis jetzt verbunden).
EPN	die Adresse des Knotens ist noch nicht gesetzt (weil er nicht verbunden ist).				
EAN	der Knoten hat eine gültige Adresse (Knoten war bis jetzt verbunden).				

Ursache / Abhilfe	Hardwareproblem: Siehe auch Kapitel Fehlersuchhilfe für VARAN-Bus-Probleme
Eintrag	VARAN,EAN:V1,D1004,P0,L1
Beschreibung	Immer in Verbindung mit „VARAN required-frame error“ oder „VARAN not-required-frame error“. Detail zum VARAN-Modul, an dem ein Fehler aufgetreten ist. <ul style="list-style-type: none"> D DeviceID L Hierarchieebene im VARAN Netzwerk P Port Nummer V VendorID
Ursache / Abhilfe	Hardwareproblem: Siehe auch Kapitel Fehlersuchhilfe für VARAN-Bus-Probleme
Eintrag	VARAN;1;VM0;iSPIMasterReadList;DOL Header;iRet=-26 VARAN;1;VM0;iSPIMasterProcessIDs;List Header,PageAddr=0x0007FF84,ListID=4 VARAN;1;VM0;iSPIMasterProcessIDs;List Header;iRet=-26 VARAN;1;VM0;iSPIMasterGetIDs;Process IDs;iRet=-26 VARAN;1;VM0;iConnectDevice;Address=0x00050000 VARAN;1;VM0;iConnectDevice;Tree:02 VARAN;1;VM0;Error no connection, VID=1,DID=1072,SN=01933607,iRet=-26 VARAN;1;VM0;Tree Topology: 0
Beschreibung	Fehler während der Konfigurationsphase des Moduls Beschreibung: „iConnectDevice;Tree:02“ Modul ist angesteckt am Manager auf Port 0, am folgenden Device auf Port 2. Hinweis: Port 0 im Logfile entspricht in Hardware VARAN 1, Port 1 im Logfile entspricht in Hardware VARAN 2,...
Ursache / Abhilfe	Modul wurde nur kurz angesteckt. Hardwareproblem Siehe auch Kapitel Fehlersuchhilfe für VARAN-Bus-Probleme

8.2.6 FTP

Eintrag	FTPSVR;3;Initializing FTP-Server interface 01.01.003
Beschreibung	FTP Server wurde gestartet (Server Version 01.01.003)
Ursache / Abhilfe	INFORMATION

Eintrag	FTPSVR;1;START:Read data from C:\ftps.ini failed, rc=-1008
Beschreibung	Initialisierungsdatei C:\ftps.ini nicht vorhanden
Ursache / Abhilfe	Datei ftps.ini ins Verzeichnis C:\ kopieren.

8.2.7 VNC

Entry	VNCNSVR;1; VNCsvr: Created VNC server task!
Beschreibung	Fernbedienung / Ferndiagnose wurde gestartet
Ursache / Abhilfe	Info

8.2.8 Log19

Eintrag	Loader V02.02.092
Beschreibung	Versionsnummer des Loaders
Eintrag	Default value defines SRAM format 1
Beschreibung	Diese Meldung zeigt an, dass aufgrund des Standardwertes das SRAM-Format 2 ausgewählt wurde.
Eintrag	Platform 32 defines SRAM format 2
Beschreibung	Diese Meldung zeigt an, dass aufgrund der Plattform mit der Identifikationsnummer 32 das SRAM Format 2 ausgewählt wurde. Es gibt 2 verschiedene SRAM-Formate: <ul style="list-style-type: none">• Im Format 1 sind die nullspannungssicheren Objekte alle im nullspannungssicheren SRAM• Im Format 2 sind die unveränderlichen Daten von nullspannungssicheren Objekten in der Datei C:\RAMFILE.DAT und die veränderlichen Daten von nullspannungssicheren Objekten befinden sich im nullspannungssicheren SRAM
Eintrag	Checking Sram
Beschreibung	Informiert, dass die Konsistenzprüfung der Datenstruktur im nullspannungssicheren Datenbereich durchgeführt wird.
Eintrag	Reorganisieren des nullspannungssicheren Datenbereichs

Beschreibung	Um zu verhindern, dass sich im nullspannungssicheren Datenbereich Objekte aus verschiedenen Projekten befinden und nicht mehr gelöscht werden, wird dieser Datenbereich reorganisiert. Dabei werden vorerst alle Objekte aus diesem Datenbereich in eine Datei geschrieben (Kopie). Danach werden alle Objekte aus dem Datenbereich gelöscht. Bei der Initialisierung der nullspannungssicheren Datenobjekte des Projekts wird ein neuer Eintrag in der Datenstruktur des nullspannungssicheren Datenbereichs erstellt. Der Wert des nullspannungssicheren Datenobjekts wird dann von der Kopie übernommen, falls er sich in der Kopie befindet, andernfalls wird der Initwert aus dem Projekt übernommen. Damit erreicht man, dass nach der Projektinitialisierung im nullspannungssicheren Datenbereich nur mehr Objekte aus dem aktuellen Projekt vorhanden sind.
Eintrag	MakeSramKopie: header written to C:\LSLDATA\SRAM.CPY (20 bytes)
Beschreibung	Information, dass 20 Bytes des Headers in der Kopie-Datei der nullspannungssicheren Daten geschrieben wurden.
Eintrag	MakeSramKopie: no copy created, because sram is not valid; trying to load old copy
Beschreibung	Es wurde keine Kopie-Datei der nullspannungssicheren Daten geschrieben, da sie nicht gültig ist. Es wird versucht, eine bestehende Kopie-Datei der nullspannungssicheren Daten einzulesen.
Eintrag	LoadSramKopie: header read from C:\LSLDATA\SRAM.CPY (20 bytes)
Beschreibung	Information, dass 20 Bytes des Headers aus der Kopie-Datei der nullspannungssicheren Daten gelesen wurden.
Eintrag	MakeSramKopie: data written to C:\LSLDATA\SRAM.CPY (72 bytes)
Beschreibung	Information, dass 72 Bytes des Datenteils in der Kopie-Datei der nullspannungssicheren Daten geschrieben wurden.
Eintrag	LoadSramKopie: data read from C:\LSLDATA\SRAM.CPY (8096 bytes)
Beschreibung	Information, dass 8096 Bytes des Datenteils aus der Kopie-Datei der nullspannungssicheren Daten gelesen wurden.
Eintrag	MakeSramKopie: no need to write to C:\LSLDATA\RAMFILE.CPY (crc32=0x00000000 and udEntries=0 are still the same)
Beschreibung	Information, dass die Kopie der Datei RAMFILE.DAT nicht neu erstellt werden musste, da die bestehende Kopie die gleiche Checksumme aufweist.
Eintrag	MakeSramKopie succeeded, marking sram as invalid
Beschreibung	Information, dass die Kopie-Datei der nullspannungssicheren Daten erfolgreich geschrieben und der Inhalt des nullspannungssicheren Datenbereichs als ungültig markiert wurde.
Eintrag	Info: re-using binary descr.block-info from prev.run

Beschreibung	Information, dass die Datenstruktur der Deskriptor-Blöcke von Klassen und Objekten nicht neu erstellt werden muss. Sie kann stattdessen aus einer Datei, welche im vorherigen Projektauf erstellt wurde, wiederhergestellt werden.
Eintrag	SramSaveFile: no need to write to ramfile.dat (crc32=0x40AE06DF and udEntries=0 are still the same)
Beschreibung	Information, dass die Datei RAMFILE.DAT nicht neu erstellt werden musste, da die bestehende Datei die gleiche Checksumme aufweist.
Eintrag	Sram is now valid.
Beschreibung	Die Datenstruktur des nullspannungssicheren Datenbereichs ist jetzt gültig.
Eintrag	Sram-cells - found in copy:0, found in sram:0, not found: 0
Beschreibung	Information, woher die Elemente für die Datenstruktur der nullspannungssicheren Daten (Sram-Zellen) beim Aufbau geladen wurden.
Eintrag	08/12/16;09:11:36.843;0000000000-SramError, errorCode=00000002 08/12/16;09:11:36.843;0000000000 Version = 00000000 08/12/16;09:11:36.843;0000000000 DataStart = 03BF0040 08/12/16;09:11:36.843;0000000000 DataLength = 000FFFFC0 08/12/16;09:11:36.843;0000000005 UsedData = 00000000 08/12/16;09:11:36.843;0000000005 DataValid = 00000000 08/12/16;09:11:36.843;0000000005 udEntries = 00000000 08/12/16;09:11:36.843;0000000005 udChk = FFFFFFFF
Beschreibung	Bei der Konsistenzprüfung der Datenstruktur des nullspannungssicheren Datenbereichs wurde ein Fehler festgestellt. Alle nullspannungssicheren Datenobjekte werden mit dem Inhalt einer zuvor erstellten Kopie geladen (falls eine solche Kopie vorhanden ist) oder sie werden mit dem im Projekt festgelegten Initwert geladen.
Eintrag	LoadSramKopie failed, rc=-3, errCode = 0x2C
Beschreibung	Das Laden der Datei die beim SRAM reorganisieren als Zwischenspeicher verwendet wird, hat nicht funktioniert.

8.3 CPU-Status und Fehlermeldungen

Die Anzeige der Status- und Fehlermeldungen erfolgt im Statustest der LASAL-Class-Software. Handelt es sich bei dem Gerät um eine CPU mit Statusdisplay (7 Segmentanzeige), so wird die Status- bzw. Fehlernummer dort ebenso angezeigt.



Weiters werden die Status- bzw. Fehlernummer zusätzlich am Bildschirm des Terminals in Klartext angezeigt.

```
Command Line Interface - Sigmatek GmbH & CoKG           v01.02.053
C:\> _

Online
CAN: St00,Bd0          IP1: 10.100.100.69      Link: 100Mbps Full Duplex & 2
COM: COM1 57600 Baud    IP2: No IP address      Client connected

Status
MAY 13 2009,13:22:11      UTI011: RESET          CPU:  6.5% &
```

Nummer	Meldung	Bedeutung	Ursache/Abhilfe
00	RUN RAM	Das Anwenderprogramm wird momentan im RAM ausgeführt.	Info

		Das Display wird nicht beeinflusst.	
01	RUN ROM	<p>Das Anwenderprogramm, das im Programmspeichermodul steht, wurde in den RAM geladen und wird momentan ausgeführt.</p> <p>Das Display wird nicht beeinflusst.</p>	Info
02	RUNTIME	<p>Gesamtdauer aller zyklischer Objekte überschreitet maximale Zeit; Zeit kann durch 2 Systemvariablen konfiguriert werden:</p> <ul style="list-style-type: none"> Runtime: Verbleibende Restzeit SWRuntime: Vorwahlwert für Runtime-Zähler 	<p>Zyklischen Task der Applikation optimieren.</p> <p>Leistungsstärkere CPU verwenden</p> <p>Vorwahlwert konfigurieren.</p>
03	POINTER	Vor Ausführung des Anwenderprogramms wurden fehlerhafte Programmzeiger festgestellt.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none"> Programmspeichermodul fehlt, ist nicht programmiert oder defekt. Programm im Anwenderprogrammspeicher (RAM) ist nicht lauffähig. Batteriepufferung ausgefallen. Softwarefehler der das Anwenderprogramm überschreibt. <p>Abhilfe</p> <ul style="list-style-type: none"> Programmspeichermodul neu programmieren, im Wiederholungsfall austauschen. Pufferbatterie austauschen. Programmfehler beheben.
04	CHKSUM	Vor Ausführung des Anwenderprogramms wurde eine falsche Prüfsumme (Checksum) festgestellt.	Ursachen/Abhilfe: s. POINTER
05	WATCHDOG	Das Programm wurde durch die Watchdoglogik abgebrochen.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none"> Interrupts vom Anwenderprogramm längere Zeit gesperrt (Befehl STI vergessen). Fehlerhafte Programmierung eines Hardware-Interrupts.

			<ul style="list-style-type: none"> • Befehle INB, OUTB, INW, OUTW falsch verwendet. • Prozessor defekt • Abhilfe • Programmfehler beheben • CPU austauschen
06	GENERAL ERROR	GENERAL ERROR Das Anhalten der Applikation über die Online Schnittstelle ist fehlgeschlagen.	Dieser Fehler tritt nur im Rahmen der Betriebssystementwicklung auf.
07	PROM DEFECT	Beim Programmieren des Programmspeichermoduls ist ein Fehler aufgetreten.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none"> • Programmspeichermodul ist defekt. • Anwenderprogramm ist zu groß. • Programmspeichermodul fehlt. <p>Abhilfe</p> <ul style="list-style-type: none"> • Programmspeichermodul tauschen
08	RESET	Die CPU hat den Befehl RESET erhalten und wartet auf weitere Befehle. Das Anwenderprogramm wird nicht bearbeitet.	Info
09	WD DEFEKT	Die Hardwareüberwachungsschaltung (Watchdoglogik) ist defekt. Die CPU überprüft nach dem Einschalten die Funktionen der Watchdoglogik. Tritt bei dieser Prüfung ein Fehler auf, läuft die CPU in einer gewollten Endlosschleife, aus der sie keine Befehle mehr annimmt.	<p>Abhilfe</p> <ul style="list-style-type: none"> • CPU austauschen.
10	STOP	Die Programmausführung wurde vom Programmiersystem angehalten.	
11	PROG BUSY	Reserviert	
12	PROGRAM LENGTH	Reserviert	
13	PROG END	Das Programmieren eines Programmspeichermoduls wurde erfolgreich beendet.	Info
14	PROG MEMO	Die CPU programmiert gerade das Programmspeichermodul.	Info

15	STOP BRKPT	Die CPU wurde durch einen Breakpoint im Programm angehalten.	Info
16	CPU STOP	Die CPU wurde durch die Programmier-Software angehalten.	Info
17	INT ERROR	Die CPU hat einen falschen Interrupt ausgeführt und das Anwenderprogramm abgebrochen, oder ist auf einen unbekannten Befehl während der Ausführung des Programms gestoßen.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none"> • Ein nicht existierender Betriebssystembefehl wurde verwendet. • Stackfehler (ungleiche Anzahl von PUSH- und POP-Befehlen). • Das Anwenderprogramm wurde durch einen Softwarefehler abgebrochen. <p>Abhilfe</p> <ul style="list-style-type: none"> • Programmfehler beheben
18	SINGLE STEP	Die CPU ist im SINGLE STEP-Mode und wartet auf weitere Befehle.	Info
19	READY	An die CPU wurde ein Modul bzw. Projekt gesendet und sie ist nun bereit zum Ausführen des Programms.	Info
20	LOAD	Die Programmbehandlung ist angehalten und die CPU empfängt gerade ein Modul bzw. Projekt.	Info
21	UNZUL. MODUL	Die CPU hat ein Modul erhalten das nicht zum Projekt gehört.	<p>Abhilfe</p> <ul style="list-style-type: none"> • Projekt neu kompilieren und ganzes Projekt übertragen
22	MEMORY FULL	Der Betriebssystemspeicher (Heap) ist zu klein. Beim Aufruf einer internen Funktion oder einer Schnittstellenfunktion aus der Anwendung konnte kein Speicher mehr reserviert werden.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none"> • Es wird immer nur Speicher allokiert aber nie freigegeben. <p>Abhilfe</p> <ul style="list-style-type: none"> • Speicher freigeben
23	NOT LINKED	Beim Starten der CPU wurde festgestellt, dass ein Modul im Projekt fehlt, oder ein Modul nicht zum Projekt gehört.	<p>Abhilfe</p> <ul style="list-style-type: none"> • Projekt neu kompilieren und ganzes Projekt übertragen
24	DIV BY 0	Bei einer Division ist ein Fehler aufgetreten.	<p>Mögliche Ursachen</p> <ul style="list-style-type: none"> • Division mit 0 • Ergebnis der Division passt nicht in das Ergebnisregister.

			Abhilfe • Programmfehler beheben
25	DIAS ERROR	Beim Zugriff auf ein DIAS-Modul ist ein Fehler aufgetreten.	Hardwareproblem: Siehe auch Kapitel Fehlersuchhilfe für DIAS-Bus-Probleme
26	WAIT	CPU ist beschäftigt.	Info
27	OP PROG	Betriebssystem wird neu programmiert.	Info
28	OP INSTALLED	Betriebssystem ist neu installiert.	Info
29	OS TOO LONG	Betriebssystem kann nicht übertragen werden; Speicher zu wenig.	Neustart, Meldung an SIGMATEK
30	NO OPERATING SYSTEM	Bootloadermeldung Kein Betriebssystem im RAM gefunden.	Neustart, Meldung an SIGMATEK
31	SEARCH FOR OS	Bootloader sucht Betriebssystem im RAM.	Neustart, Meldung an SIGMATEK
32	NO DEVICE	Reserviert	
33	UNUSED CODE	Reserviert	
34	MEM ERROR	Das eingespielte Betriebssystem entspricht nicht der Hardwarekonfiguration.	Richtiges Betriebssystem verwenden
35	MAX IO	Reserviert	
36	MODULE LOAD ERROR	LASAL Modul oder Projekt konnte nicht geladen werden.	Abhilfe • Projekt neu kompilieren und ganzes Projekt übertragen
37	BOOTIMAGE FAILURE	Genereller Fehler beim Laden des Betriebssystems.	SIGMATEK kontaktieren
38	APPMEM ERROR	Fehler bei der dynamischen Applikation-Speicher-Verwaltung (Anwender-Heap).	Abhilfe • Fehler bei den allokierten Speicherzugriffen beheben
39	OFFLINE	Dieser Fehler tritt in der Steuerung nicht auf.	Dieser Fehler-Code wird im Programmiersystem benutzt um anzuzeigen, dass keine Verbindung zur Steuerung besteht.
40	APPL LOAD	Reserviert	
41	APPL SAVE	Reserviert	

42	ETHERCAT MANAGER ERR	Im EtherCAT Manager wurde eine Fehlernummer hinterlegt und die Programmausführung angehalten.	
43	ETHERCAT ERROR	Ein benötigter EtherCat-Client wurde abgesteckt oder es trat ein Kommunikationsfehler auf.	
44	VARAN MANAGER ERROR	Im VARAN Manager wurde eine Fehlernummer hinterlegt und die Programmausführung angehalten.	Abhilfe <ul style="list-style-type: none"> Logfile lesen
45	VARAN ERROR	Ein benötigter VARAN-Client wurde abgesteckt oder es trat ein Kommunikationsfehler mit einem VARAN-Client auf.	Abhilfe <ul style="list-style-type: none"> Logfile lesen Error Tree
46	APPL-LOAD-ERROR	Fehler beim Laden der Applikation.	Mögliche Ursachen <ul style="list-style-type: none"> Applikation wurde gelöscht. Abhilfe <ul style="list-style-type: none"> Applikation neu zur Steuerung übertragen.
47	APPL- SAVE- ERROR	Fehler beim Speichern der Applikation.	
50	ACCESS- EXCEPTION-ERROR	Lese-Schreibzugriff auf unerlaubtem Speicherbereich.(z.B. Schreiben auf NULL-Pointer).	Abhilfe <ul style="list-style-type: none"> Applikationsfehler beheben
51	BOUND EXCEEDED	Exception-Fehler bei Zugriff auf Arrays. Speicherbereichsüberschreitung in Form eines Zugriffs auf ein ungültiges Element.	Abhilfe <ul style="list-style-type: none"> Applikationsfehler beheben
52	PRIVILEGED INSTRUCTION	Unerlaubter Befehl für aktuellen CPU-Level, z.B. setzen der Segment-Register.	Mögliche Ursachen <ul style="list-style-type: none"> Programmcode der Applikation wurde von der Applikation überschreiben. Abhilfe <ul style="list-style-type: none"> Applikationsfehler beheben
53	FLOATING POINT ERROR	Fehler während einer Gleitkomma-Operation.	
60	DIAS-RISC-ERROR	Error vom intelligenten DIAS-Master.	Neustart, Meldung an SIGMATEK
64	INTERNAL ERROR	Interner Fehler, alle Applikationen gestoppt.	Neustart, Meldung an SIGMATEK
65	FILE ERROR	Fehler während Dateioperation.	

66	DEBUG ASSERTION FAILED	INTERNAL ERROR	Neustart, Meldung an SIGMATEK
67	REALTIME RUNTIME	Gesamtdauer aller Realtime-Objekte überschreitet maximale Zeit; Zeit kann nicht konfiguriert werden. 2 ms bei 386er CPUs 1 ms bei restlichen CPUs	Abhilfe <ul style="list-style-type: none"> • Echtzeit Task der Applikation optimieren (RtWork). • Echtzeit Task Taktzeit aller Objekte verlangsamen. • Applikationsfehler beheben • CPU ist im Realtime zu ausgelastet => Leistungsstärkere CPU verwenden.
68	BACKGROUND RUNTIME	Gesamtdauer aller Background Objekte überschreitet maximale Zeit; Zeit kann durch 2 Systemvariablen konfiguriert werden: -BTRuntime: Verbleibende Restzeit -SWBTRuntime: Vorwahlwert für Runtime-Zähler	Abhilfe <ul style="list-style-type: none"> • Background Task der Applikation optimieren (Background) • Leistungsstärkere CPU verwenden • SWBTRuntime richtig einstellen
72	S-DIAS ERROR	Es ist ein Fehlerfall in Verbindung mit einem S-DIAS-Modul aufgetreten.	Mögliche Ursachen <ul style="list-style-type: none"> • reales Netzwerk stimmt nicht mit Projekt überein • S-DIAS Client ist defekt Abhilfe <ul style="list-style-type: none"> • Logfile auswerten
95	USER DEFINED 0	Frei verwendbarer Code	
96	USER DEFINED 1	Frei verwendbarer Code	
97	USER DEFINED 2	Frei verwendbarer Code	
98	USER DEFINED 3	Frei verwendbarer Code	
99	USER DEFINED 4	Frei verwendbarer Code	
100	C_INIT	Start der Initialisierung, Konfiguration wird durchgeführt.	
101	C_RUNRAM	LASAL Projekt wurde erfolgreich vom RAM gestartet.	
102	C_RUNROM	LASAL Projekt wurde erfolgreich vom ROM gestartet.	

103	C_RUNTIME		
104	C_READY	Alles in Ordnung	
105	C_OK	Alles in Ordnung	
106	C_UNKNOWN_CID	Unbekannte Klasse von einem stand-alone oder embedded Objekt; oder unbekannte Basis-Klasse.	
107	C_UNKNOWN_CONSTR	Betriebssystemklasse kann nicht erstellt werden, wahrscheinlich falsches Betriebssystem.	
108	C_UNKNOWN_OBJECT	Hinweis auf ein unbekanntes Objekt in einem Interpreter Programm; Erstellung von mehr als einem DCC080-Objekt;	
109	C_UNKNOWN_CHNL	Nummer des HW-Moduls größer als 60.	
110	C_WRONG_CONNECTION	Keine Verbindung zu erforderlichen Kanälen.	
111	C_WRONG_ATTR	Falsche Server-Attribute.	
112	C_SYNTAX_ERROR	Kein spezifizierter Fehler, alle Teilprojekte neu kompilieren, alles übertragen.	
113	C_NO_FILE_OPEN	Versuchte eine unbekannte Tabelle zu öffnen.	
114	C_OUTOF_NEAR	Speicherzuteilung fehlgeschlagen.	
115	C_OUT_OF_FAR	Speicherzuteilung fehlgeschlagen.	
116	C_INCOMPATIBLE	Objekt mit gleichem Namen existiert bereits, hat aber eine andere Klasse.	
117	C_COMPATIBLE	Objekt mit dem selben Namen und der selben Klasse existiert bereits, muss upgedated werden.	
224	LINKING	Applikation wird gelinkt.	
225	LINKING ERROR	Fehler beim Linken, Meldung im LASAL Status-Fenster.	
226	LINKING DONE	Linken beendet	
230	OP BURN	Betriebssystem wird in den Flashspeicher gebrannt.	
231	OP BURN FAIL	Fehler beim Brennen des Betriebssystems.	

232	OP INSTALL	Betriebssystem wird installiert.	
240	USV-WAIT	Versorgung wurde abgeschaltet, USV ist aktiv. System wird heruntergefahren	
241	REBOOT	Betriebssystem wird neu gestartet.	
242	LSL SAVE		
243	LSL LOAD		
252	CONTINUE		
253	PRERUN	Applikation wird gestartet.	
254	PRERESET	Applikation wird beendet.	
255	CONNECTION BREAK		

8.3.1 Fehlersuche für DIAS-Bus-Probleme

Genauen Wert der 24 V nachgemessen?



Alle Komponenten versorgt (DC OK LED oder nachmessen)?



Bei allen Komponenten leuchtet oder blinkt die RX und TX LED bei laufender Applikation (falls vorhanden)?



Bei allen Komponenten leuchtet das Synchron LED bei laufender Applikation (falls vorhanden)?





Leuchtet während laufender Applikation das Reset LED (falls vorhanden, z.B. CIC)?



Retries DIASMaster, DIV512, CIV521, ... ?



Retries nur wenn Frequenzumrichter oder andere Störquelle läuft?



Abschlusswiderstände richtig gesetzt?



Abschlusswiderstände doppelt gesetzt?



Stationsnummern doppelt vergeben?



Bei verschiedenen Modulen sind LOW und HIGH Hexschalter vertauscht
=> Auf die Beschriftung achten



Bei Repeatern gibt es einen Eingang und einen Ausgang (CIC012,DIC121,...!)



Kabellängen überschritten?

Max. 20 m ohne Repeater mit Lappkabel / UNITRONIC BUS FD P LD 2 x 2 x 0.25 mm²



Kabeltyp geeignet?



Mit welchem DIAS-Error stürzt die CPU ab (CPU Display oder 7 Segment)?

Bei 7 Segment auf Set-Taste drücken, um die Stationsnummer anzuzeigen?



Wie sind die Erdungsmaßnahmen/Schirmauflagen/Schaltschrankaufbauten?



Alle Komponenten geerdet (Modulträger, Hutschiene, CPU, ...)?



Werden alle Komponenten in der Software erkannt?



Stimmt die ausgelesene Hardwarekonfiguration mit der tatsächlichen Software Konfiguration überein?



Spannungseinbrüche am Versorgungsmodul des DIAS-Busses?



Modul defekt?

8.4 Fehlersuche für VARAN-Bus-Probleme

8.4.1 Error Tree

Bei CPUs mit Display wird bei neueren Betriebssystemen bei einem VARAN-Error automatisch der VARAN Error Tree aufgerufen. Mit diesem Error Tree kann bestimmt werden welches Modul den Fehler verursachte. Bei manchen Betriebssystemen muss vorher noch die Taste T gedrückt werden.

```
Command Line Interface - Sigmatek GmbH & CoKG          v01.02.053
* * 0* 1* 2* 3* 4* 5* 6* 7* 8* 9* 10* 11* 12* 13* 14* 15* 16* 17* 18
1  M0:VMC052
2  M0--00:VTI011
3  E M0--00:00:nc
4  M0--01:nc
Quit:ESC,Q  Err-Log:E  SaveTree:S  Navigation:Enter,Backspace,U,D,L,R,0,9

Online
CAN: S000_B10 Baud          IP1: 10.100.100.62  Link: 100Mbps Full Duplex & 1
COM: COM1 57600 Baud        IP2: No IP address  Client connected

Status
MAY 13 2009,12:19:37          VTI011: VARAN ERROR(00)          CPU: 3.7% &

Autoexec  Settings  Virtual Keyboard
```

Die Zahlen 0 bis 18 und 1 bis 4 dienen nur zur Navigationshilfe bei großen VARAN-Netzwerken. Navigation mit den Tasten: Enter, Backspace, U, D, L, R, 0, 9

1· Zeile: M0:VMC052:

Manager 0 hat die Bezeichnung VMC052 (könnte auch: ETV1961-K, HGT833, ETV0811,...)

2· Zeile: M0:00:VTI011 :

Am Port 0 des Managers 0 ist ein VTI011 angesteckt.

3· Zeile: E M0 00-00 : nc

Am Port 0 des VTI011 trat ein Fehler auf (E => Error).

In diesem Fall wurde bei laufender Applikation der Client am Downlinkport des VT1011 abgesteckt.



Anhand des Error Trees kann zwar bestimmt werden welcher Zugriff auf welches Modul fehlgeschlagen ist, dies muss aber nicht heißen, dass der Fehler durch das Austauschen des Moduls behoben ist. Der Fehler kann lediglich auf die Strecke zwischen CPU und Fehlermodul eingeschränkt werden. Für diese Strecke muss nun die Hardwarecheckliste angewandt werden.

8.4.2 Hardware-Checkliste

Genauen Wert der 24 V nachgemessen?



Alle Komponenten versorgt (DC OK LED oder nachmessen)?



Bei allen Komponenten leuchtet oder blinkt Link (grünes LED) und Active (oranges LED)?



Bei allen Komponenten leuchtet das Synchron LED (falls vorhanden)?



Leuchtet während laufender Applikation das Reset LED (falls vorhanden, z.B. CIV).



 Log-Datei lesen

 Retries VARAN Manager

=> Erdungsmaßnahmen/Schirmauflagen/Schaltschrankaufbau kontrollieren.

 Kabel richtig konfektioniert?

Kabellängen müssen kleiner gleich 100 m sein

 Modulspezifische Retrycounter

Mit dem modulspezifischen Retrycounter können schlecht verlegte oder falsch verdrillte Kabel gefunden werden. Retries nur wenn Frequenzumrichter oder andere Störquelle läuft?

=> Erdungsmaßnahmen/Schirmauflagen/Schaltschrankaufbau kontrollieren.

 Mit welchem VARAN Error steigt die CPU aus (Logfile lesen, CPU-Status am Display oder 7-Segment Anzeige)?

=> Erdungsmaßnahmen/Schirmauflagen/Schaltschrankaufbau kontrollieren.

 Alle Komponenten geerdet (Modulträger, Hutschiene, CPU, ...)?

 Werden alle Komponenten erkannt?



Stimmt die ausgelesene Hardwarekonfiguration mit der tatsächlichen Software Konfiguration überein?



Spannungseinbrüche am Modul?



Modul defekt?

9 SIGMATEK Device Configuration

Es besteht die Möglichkeit, mittels eines Web-Services SIGMATEK-Steuerungen zu konfigurieren. In diesem Kapitel wird die entsprechende Vorgangsweise beschrieben.

Die SIGMATEK Device Configuration steht ausschließlich unter dem Betriebssystemen Salamander/Gecko zur Verfügung!



- [Allgemein](#)
- [Das LOGIN-Fenster](#)
- [Bereiche in der Device Configuration](#)
- [Logout](#)
- [Troubleshooting](#)
- [Disclaimer](#)

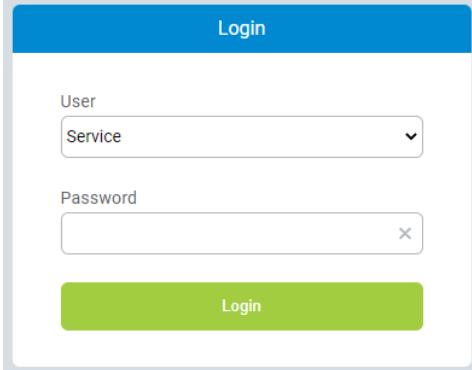
9.1 Allgemein

Eine SIGMATEK-CPU mit Salamander/Gecko Betriebssystem kann mittels Webservice konfiguriert werden. Hierzu in einem Browser die IP-Adresse der zu konfigurierenden Steuerung plus den vorkonfigurierten Port angeben. (Standardmäßig vorkonfiguriert ist es <http://10.10.150.1:1980>)

Die erstellte Anwenderkonfiguration wird auf der Steuerung als Datei C:\lslsys\webconfigusers abgelegt. So eine Konfiguration kann mittels eines USB-Sticks, oder über das Machine Manger Update Tool auf andere Steuerungen übertragen werden.

9.2 Das LOGIN-Fenster

Wird die SIGMATEK Device Configuration das erste mal aufgerufen, oder wurde ein "Reload" ausgeführt bzw. hat sich der Anwender abgemeldet, wird das Login-Fenster dargestellt:



Login

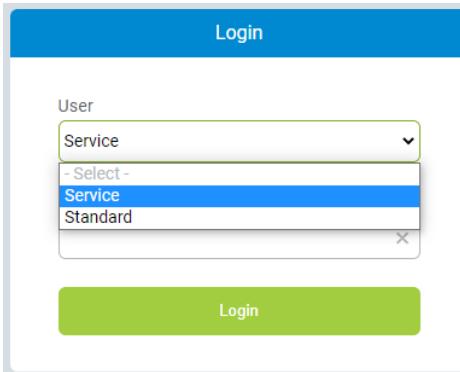
User

Service

Password

>Login

Im Login-Fenster in der SIGMATEK Device Configuration kann zwischen zwei vordefinierten Anwendern ausgewählt werden: Service und Standard.



Login

User

Service

- Select -

Service

Standard

>Login

Das Standardpasswort im Auslieferungszustand ist für beide Anwender "sigmatek".



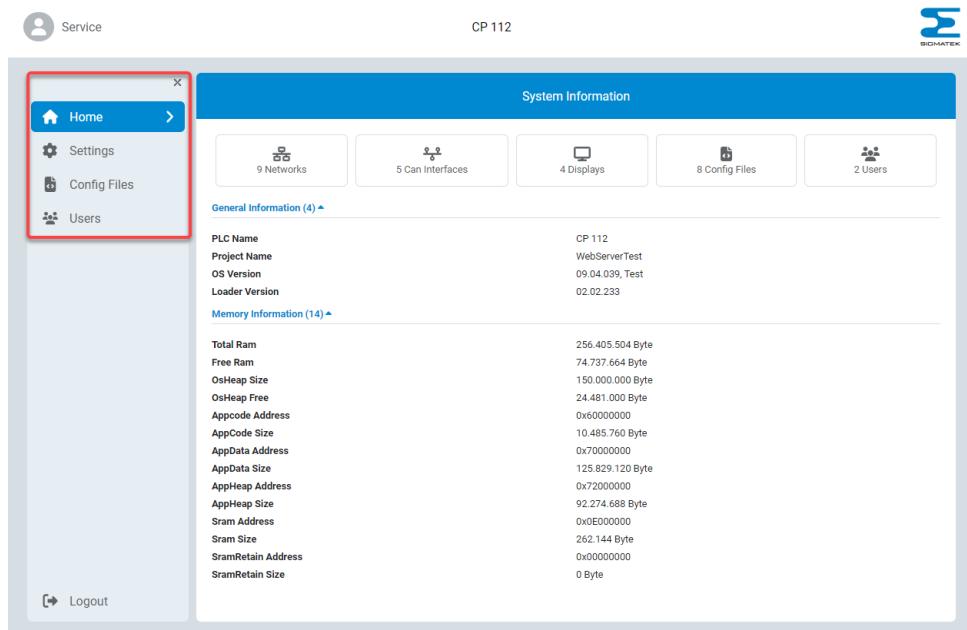
Über die Eigenschaften der Anwender siehe das Kapitel [Benutzerverwaltung](#).

Über das Ausloggen aus der SIGMATEK Device Configuration bzw. das Übernehmen der Änderungen siehe das Kapitel [Logout](#).

9.3 Bereiche in der Device Configuration

Ist man erfolgreich eingeloggt, kann es an die Konfiguration gehen. Es befinden sich auf der linken Seite maximal vier Bereiche, welche je nach Anwenderberechtigung angezeigt werden können. Ist man mit dem Anwendernamen [Service](#) angemeldet, werden alle Bereiche angezeigt; ist man mit [Standard](#) eingeloggt, so sind nur die ersten beiden Bereiche sichtbar.

Die Bereiche können je nach Bildschirmgröße- und Auflösung auf drei verschiedene Arten dargestellt werden. Bei hohen Auflösungen links als Icons mit Namen:



Service

CP 112



System Information

General Information (4) ▲

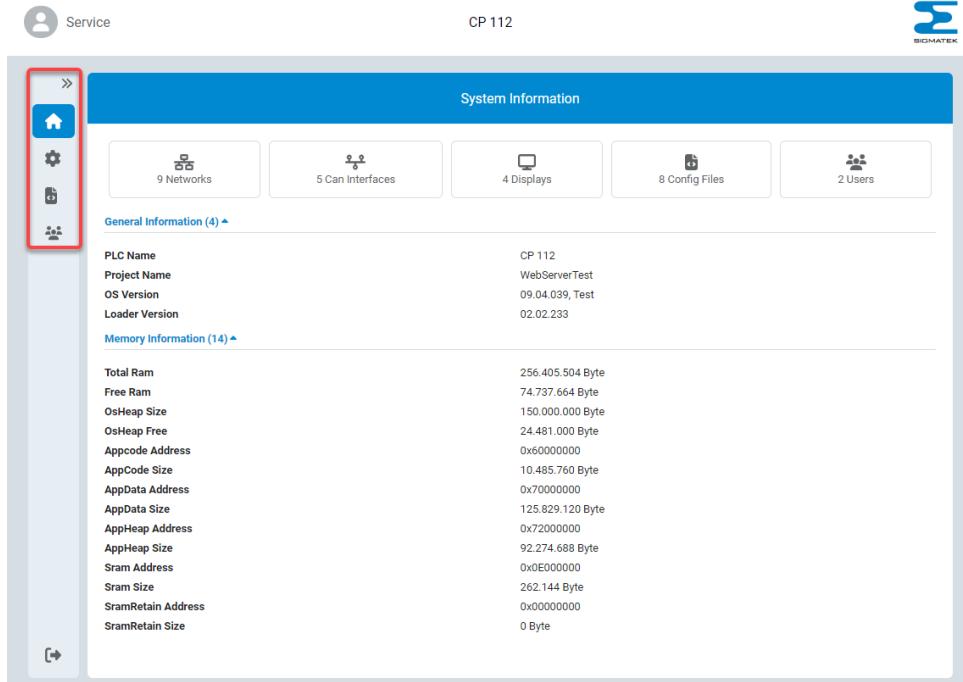
PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

Memory Information (14) ▲

Total Ram	256.405.504 Byte
Free Ram	74.737.664 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
AppCode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	125.829.120 Byte
AppHeap Address	0x72000000
AppHeap Size	92.274.688 Byte
Sram Address	0x0E000000
Sram Size	262.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

Logout

Klickt man auf das x über die Home-Taste, wird die Bereichsanzeige eingeklappt, sprich: es sind nur die Icons sichtbar, die Beschreibungen verschwinden:



Service

CP 112

System Information

9 Networks 5 Can Interfaces 4 Displays 8 Config Files 2 Users

General Information (4) ▲

PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

Memory Information (14) ▲

Total Ram	256.405.504 Byte
Free Ram	74.737.664 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
Appcode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	125.829.120 Byte
AppHeap Address	0x72000000
AppHeap Size	92.274.688 Byte
Sram Address	0x0E000000
Sram Size	262.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

Mit dem Doppelpfeil über dem Home-Icon kann die Bereichsanzeige wieder ausgeklappt werden. Ist die Darstellung - z.B. für ein Mobilgerät - in kleinerer Auflösung und Hochformat ausgerichtet, wandern die Icons zur Unterseite:

Service CP 112 SIGMATEK

System Information

9 Networks	5 Can Interfaces	4 Displays
8 Config Files	2 Users	

General Information (4) ▲

PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

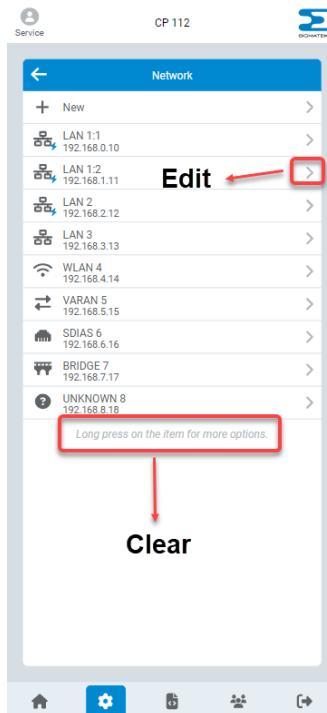
Memory Information (14) ▲

Total Ram	256.405.504 Byte
Free Ram	74.737.664 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
Appcode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	125.829.120 Byte
AppHeap Address	0x72000000
AppHeap Size	92.274.688 Byte
Sram Address	0x0E000000
Sram Size	262.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

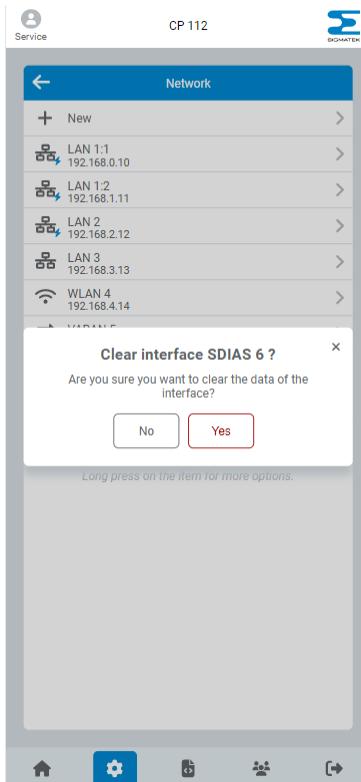
Besondere Bedienung im Hochformat

An dieser Stelle soll auf eine Besonderheit bei der Bedienung in dieser "mobilen" Hochformat-Auflösung hingewiesen werden. Wird ein Edit-Fenster (z.B. in den [Settings](#) unter [Network](#), [Can Bus](#), oder [Display](#)) in diesem Format geöffnet, können aufgrund der geringen Breite keine "Clear"- und "Edit"-Buttons dargestellt werden. Stattdessen wird in jeder dargestellten Zeile auf der rechten Seite ein More-Pfeil angezeigt sowie unten der Hinweis "Long press on the item for more options".



Ein kurzes Tippen auf den Eintrag entspricht dem "Edit"-Button in der horizontalen Auflösung.

Ein längerer Druck entspricht dem "Clear"-Button in der horizontalen Auflösung. Es wird auch gleich der Rückfragedialog des Clear-Buttons angezeigt:



Nun folgt die Beschreibung der einzelnen Bereiche:

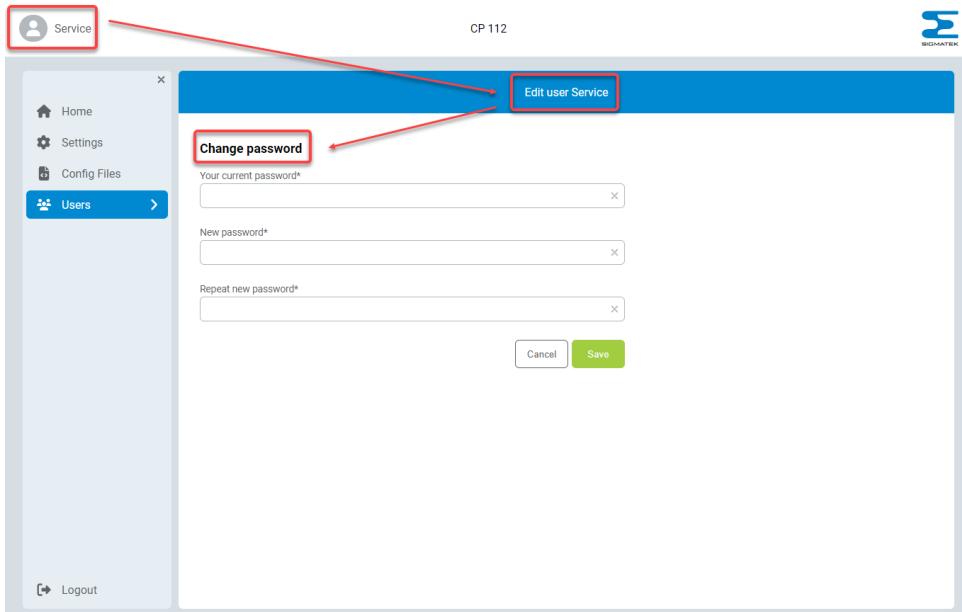
- [Titelleiste](#)
- [Home](#)
- [Settings](#)
- [Config Files](#)
- [Users](#)

9.3.1 Titelleiste

In der Titelleiste der Website sind drei Symbole zu sehen.



Links ist der gerade eingeloggte Anwender zu sehen. Mit einem Mausklick auf den eingeloggten User wird der Dialog für die (eigene) Passwortänderung angezeigt:



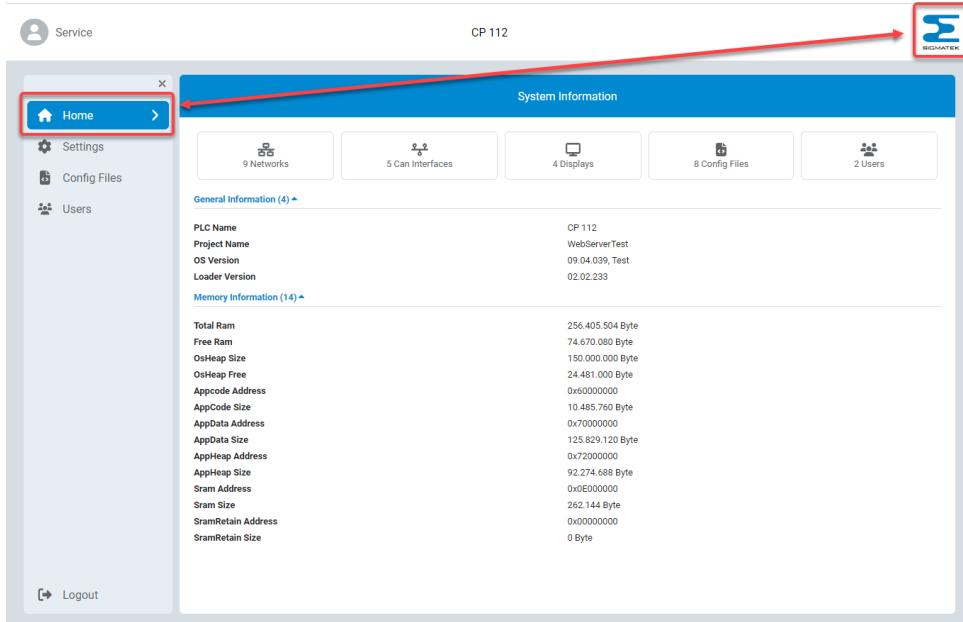
In der Mitte sieht man den Typ (Name) der Steuerung, welche konfiguriert werden soll:



Der Button rechts...



...entspricht dem Button für den Home-Bereich:



The screenshot shows the LASAL OS web interface. At the top left is a user icon labeled "Service". In the top center, the text "CP 112" is displayed. At the top right is the SIGMATEK logo. A red box highlights the "Home" button in the top navigation bar, and a red arrow points from this box to the SIGMATEK logo. The main content area is titled "System Information". It features a "General Information (4)" section with the following data:

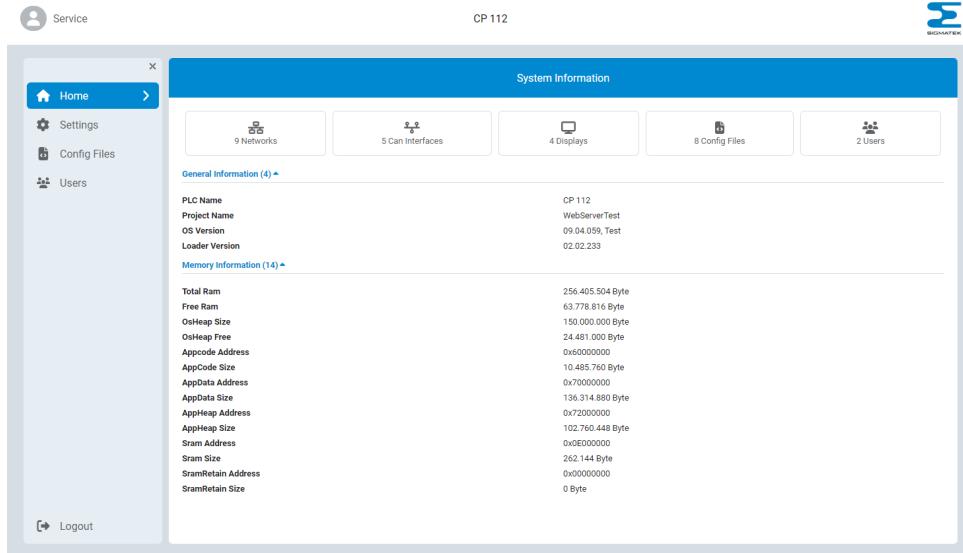
PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

Below this is a "Memory Information (14)" section with the following data:

Parameter	Value
Total Ram	256.405.504 Byte
Free Ram	74.670.080 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
Appcode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	125.829.120 Byte
AppHeap Address	0x72000000
AppHeap Size	92.274.688 Byte
Sram Address	0x0E000000
Sram Size	262.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

At the bottom left is a "Logout" button.

9.3.2 Home



The screenshot shows the LASAL OS Home interface. At the top, there is a navigation bar with icons for Home, Settings, Config Files, and Users. The main area is titled "System Information" and contains two sections: "General Information" and "Memory Information".

General Information (4) ▾

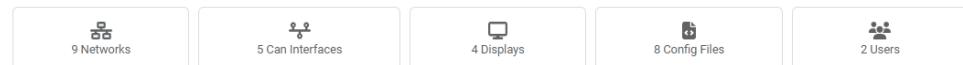
PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.099, Test
Loader Version	02.02.233

Memory Information (14) ▾

Total Ram	256.405.504 Byte
Free Ram	63.778.816 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
AppCode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	136.314.880 Byte
AppHeap Address	0x72000000
AppHeap Size	102.760.448 Byte
Sram Address	0x0E000000
Sram Size	262.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

At the bottom left is a "Logout" button.

Der Home-Bildschirm zeigt Informationen der Steuerung an, welche bei einer bestehenden Online-Verbindung auch in PlcDiag (Target Info) bzw. LASAL CLASS 2 (Projekt Info) angezeigt werden können. Außerdem gibt es im oberen Bereich fünf fixe, vorkonfigurierte Shortcuts, welche auf gewisse Unterpunkte in einzelnen Bereichen zeigen:

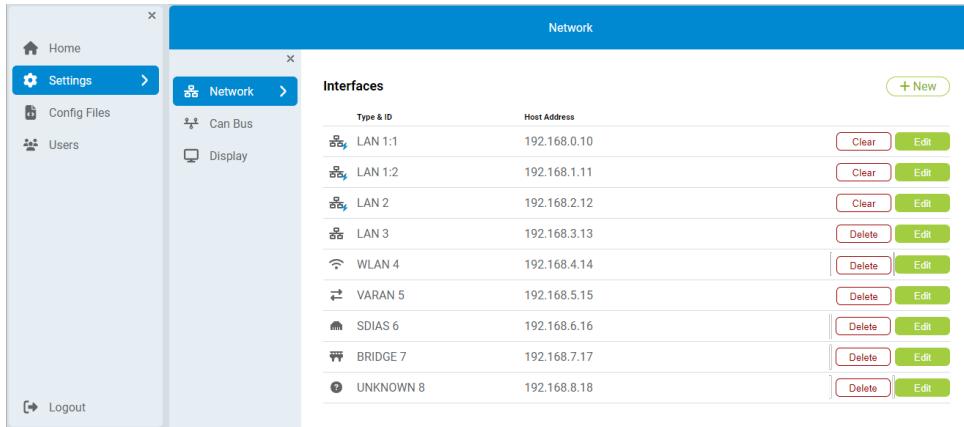


Der Anwender "Standard" sieht nur die ersten drei Shortcuts! Für nähere Informationen über die auswählbaren Anwendern siehe das Kapitel Benutzerverwaltung.

Die ersten drei Shortcuts rufen die entsprechenden Punkte des Bereiches Settings (Network, Can Bus, Display) auf.

Die letzten beiden Shortcuts rufen die beiden Bereiche (Config Files, Users), welche nur der Anwender "Service" sieht, auf.

9.3.3 Settings



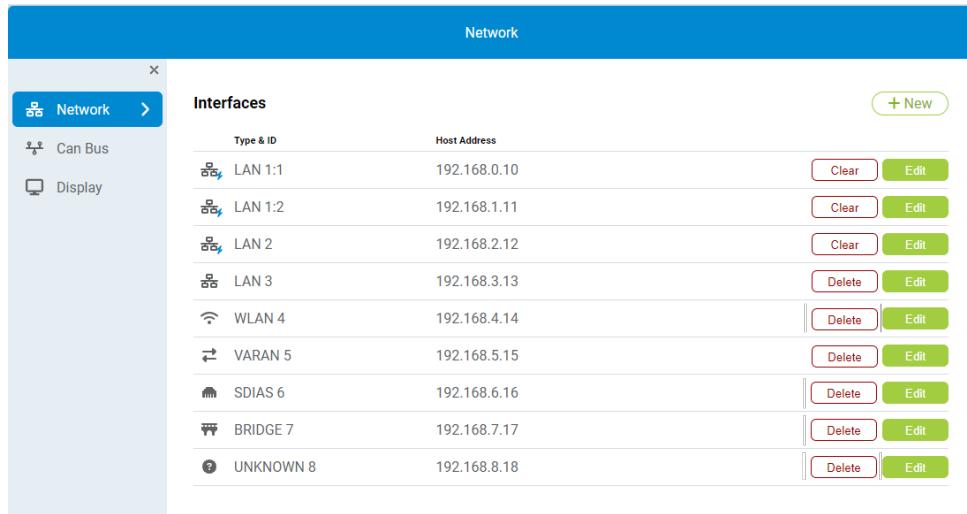
Type & ID	Host Address		
LAN 1:1	192.168.0.10	<input type="button" value="Clear"/>	<input type="button" value="Edit"/>
LAN 1:2	192.168.1.11	<input type="button" value="Clear"/>	<input type="button" value="Edit"/>
LAN 2	192.168.2.12	<input type="button" value="Clear"/>	<input type="button" value="Edit"/>
LAN 3	192.168.3.13	<input type="button" value="Delete"/>	<input type="button" value="Edit"/>
WLAN 4	192.168.4.14	<input type="button" value="Delete"/>	<input type="button" value="Edit"/>
VARAN 5	192.168.5.15	<input type="button" value="Delete"/>	<input type="button" value="Edit"/>
SDIAS 6	192.168.6.16	<input type="button" value="Delete"/>	<input type="button" value="Edit"/>
BRIDGE 7	192.168.7.17	<input type="button" value="Delete"/>	<input type="button" value="Edit"/>
UNKNOWN 8	192.168.8.18	<input type="button" value="Delete"/>	<input type="button" value="Edit"/>

Im Bereich "Settings" können drei weitere Buttons angeklickt und mit ihnen [Netzwerk](#)-, [Can-Bus](#)- sowie [Displayschnittstellen](#) konfiguriert werden. Aus dem Bereich "[Home](#)" aus weisen die ersten drei fix konfigurierten Shortcuts auch auf diese Seiten hin.



Alle Änderungen werden erst nach Neustart der Systems übernommen, welcher entweder manuell, oder im [Logout-Dialog](#) mit dem Button "Reboot" erfolgen kann.

9.3.3.1 Network



Interfaces		
Type & ID	Host Address	
LAN 1:1	192.168.0.10	Clear Edit
LAN 1:2	192.168.1.11	Clear Edit
LAN 2	192.168.2.12	Clear Edit
LAN 3	192.168.3.13	Delete Edit
WLAN 4	192.168.4.14	Delete Edit
VARAN 5	192.168.5.15	Delete Edit
SDIAS 6	192.168.6.16	Delete Edit
BRIDGE 7	192.168.7.17	Delete Edit
UNKNOWN 8	192.168.8.18	Delete Edit

Auf dieser Seite können Netzwerksschnittstellen konfiguriert und verwaltet werden. Mit dem "x" über "Network" kann diese zweite Menüebene ebenso zugeklappt und mit dem daraufhin erscheinenden Doppelpfeil aufgeklappt werden, wie es in der oberste Ebene möglich ist. In zugeklapptem Zustand werden auch hier nur mehr die Icons und keine Beschreibungen mehr dargestellt.

Eine neue Netzwerksschnittstelle kann mittels des "+New"-Buttons erstellt werden:

New network

x

 Network >

 Can Bus

 Display

+ Interface

Interface ID*

Type

use DHCP

Host Address*

Subnet*

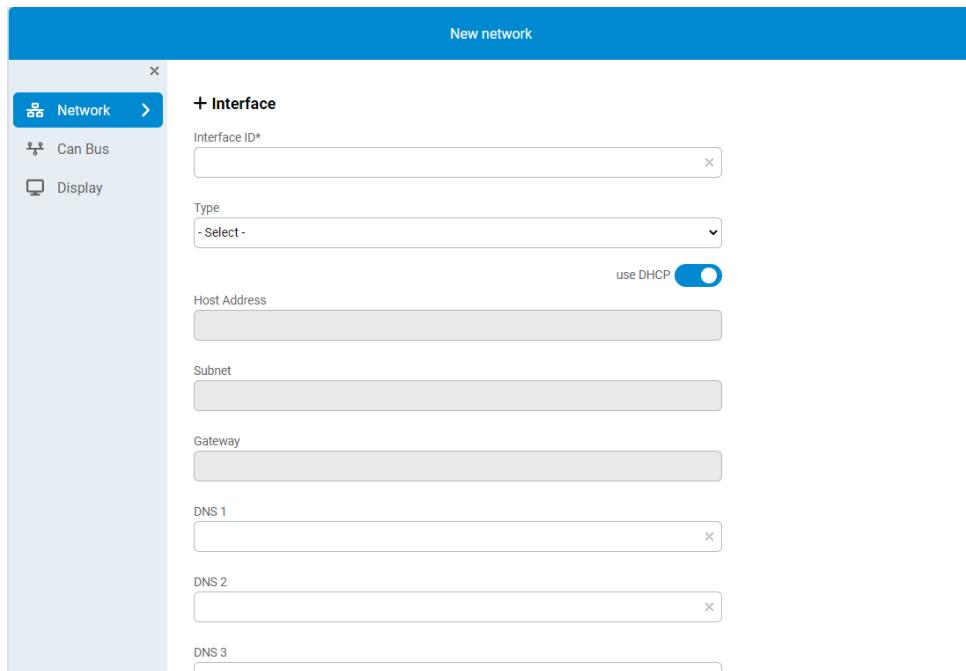
Gateway

DNS 1

DNS 2

DNS 3

Wie allgemein üblich, sind Felder, die mit einem Stern (*) gekennzeichnet sind, Pflichtfelder. Allerdings kann hier das Auswählen von "use DHCP" die Art der Felder ändern:



Wie man gut erkennen kann, sind Felder, welche in der anderen Stellung des "use DHCP"-Schalters Pflichtfelder waren, disabled (ausgegraut), und sind somit auch keine Pflichtfelder mehr.

Die ID bei Interfaces vom Typ "LAN" kann entweder einstellig, oder aus zwei Zahlen, getrennt durch einen Doppelpunkt, bestehen. Einer LAN-Schnittstelle können maximal zwei IP-Adressen zugeordnet werden, somit kann sie gleichzeitig in zwei Netzwerken registriert sein. Im ersten Bild in diesem Kapitel ist LAN 1 so eine Schnittstelle. Die erste Zahl ist die ID, die zweite die Nummer des Netzwerkes / der IP-Adresse.

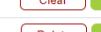
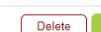
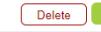
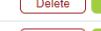
Im ersten Bild fällt auch noch auf, dass manche Icons in der Spalte "Type & ID" einen



kleinen blauen Blitz rechts unten haben (), andere nicht. Dieser blaue Blitz ist das "attached"-Symbol, bedeutet also, dass die Schnittstelle tatsächlich als Hardware vorhanden (in der CPU, oder z.B. als USB-Stick ansteckt) ist. Die Schnittstellen ohne dieses Symbol wurden zwar konfiguriert, sind jedoch physisch nicht vorhanden.

Rechts neben der "Type & ID"-Spalte wird die Hostadresse sowie zwei weitere Buttons ("Clear" oder "Delete" und "Edit") angezeigt. Mit "Clear" wird bei einer physisch tatsächlich

vorhandenen Schnittstelle (mit dem Blitz) nach einer Rückfrage die Konfiguration dieser Schnittstelle gelöscht - dies ist dann daran zu erkennen, dass das entsprechende Icon hellgrau dargestellt wird, in der Spalte "Host Address" statt einer IP-Adresse "not configured" steht -, und der "Clear"-Button verschwindet:

Network			
	Interfaces		
	Type & ID	Host Address	
 Can Bus	LAN 1:1	not configured	
 Display	LAN 1:2	192.168.1.11	 
	LAN 2	192.168.2.12	 
	LAN 3	192.168.3.13	 
	WLAN 4	192.168.4.14	 
	VARAN 5	192.168.5.15	 
	SDIAS 6	192.168.6.16	 
	BRIDGE 7	192.168.7.17	 
	UNKNOWN 8	192.168.8.18	 

Bei physisch nicht vorhandenen Schnittstellen (ohne Blitz) gibt es keinen "Clear"-, sondern einen "Delete"-Button. Mit einem Klick auf diesen wird die Schnittstelle ganz entfernt.

Mit dem Button "Edit" kann die jeweilige Netzwerksschnittstelle konfiguriert werden. Es wird genau das gleiche Fenster angezeigt, wie beim Klicken auf den "+New"-Button, nur sind die Felder bereits mit den aktuellen Daten befüllt.

Die ersten beiden Feldern (Interface ID und Type) können bei "attached"-Schnittstellen nicht geändert werden. Ebenso wenig kann in so einem Fall das letzte Feld ("Hardware Address") geändert werden. Weiter oben wurde schon erwähnt, dass - die drei - Felder unterhalb des "use DHCP"-Schalters disabled werden, sobald dieser eingeschaltet wird.

Mit "Cancel" kann der Konfigurationsvorgang abgebrochen werden, mit "Save" werden die Änderungen übernommen.



Wird auf "Save" geklickt, werden die gewählten Einstellungen in die Datei Autoexec.lsl zurückgeschrieben, es soll also mit Bedacht konfiguriert werden, denn es ist möglich, dass nach einer Änderung keine Verbindung zu der Steuerung mehr möglich ist!



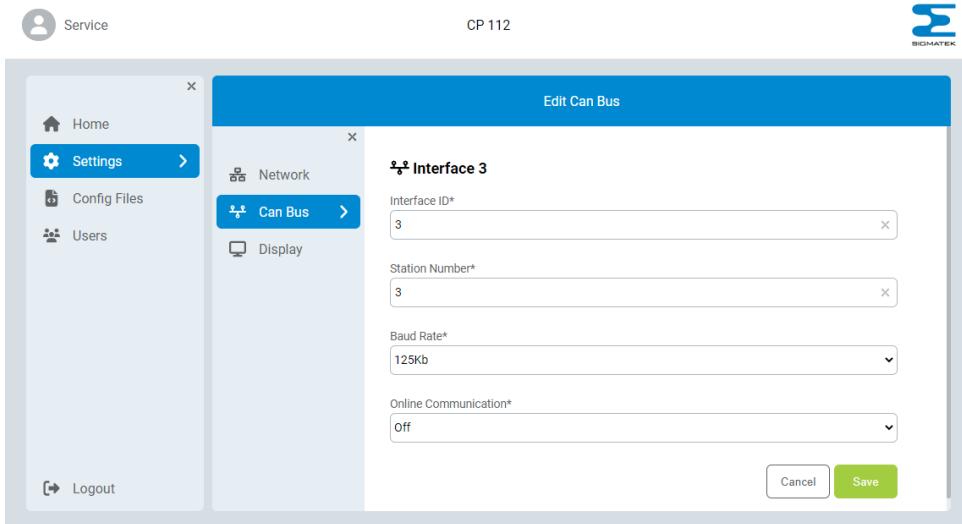
Alle Änderungen werden erst nach Neustart der Systems übernommen, welcher entweder manuell, oder im [Logout-Dialog](#) mit dem Button "Reboot" erfolgen kann.

9.3.3.2 CAN-Bus

Das Verwaltungsfenster für die Konfiguration von Can Bus-Schnittstellen ist dem für Netzwerksschnittstellen sehr ähnlich.

ID	Station & Baud Rate		
CAN 1	Station 2 / 250kb	Clear	Edit
CAN 3	Station 7 / 500kb	Delete	Edit
CAN 5	Station 2 / 100kb	Delete	Edit

Die Anzeige ("blauer Blitz" für tatsächlich vorhandene, kein Blitz für vorkonfigurierte, aber physisch nicht vorhandene Hardware) so wie die Funktionsweise der Buttons "+New", "Clear" bzw. "Delete" und "Edit" sind mit denen in der Verwaltung der Netzwerksschnittstellen identisch. Klarerweise unterscheidet sich die 2. Spalte (Stationsnummer und Baudrate) so wie die eingebbaren Felder, wenn man auf "Edit" klickt:



Service CP 112

Edit Can Bus

Interface 3

Interface ID*
3

Station Number*
3

Baud Rate*
125kb

Online Communication*
off

Cancel Save

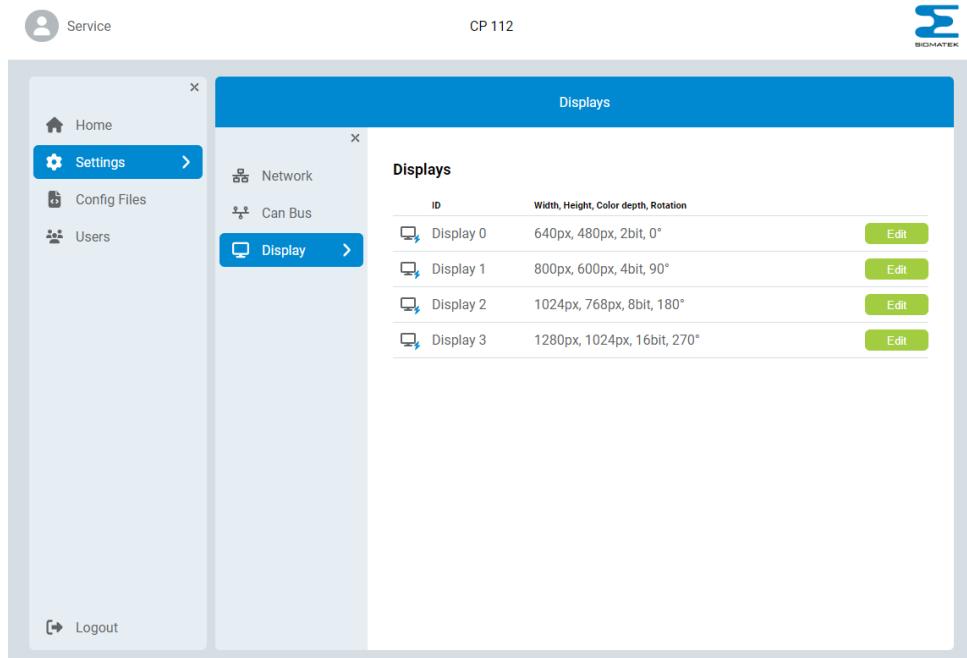
ID, Stationsnummer und Baudrate sind selbsterklärend. Das letzte Feld "Online Communication" bestimmt, ob man durch diese Can Bus-Schnittstelle von LASAL aus Online gehen können soll (On), oder nicht (Off). Auch auf dieser Seite sind die mit einem Stern (*) gekennzeichneten Felder Pflichtfelder.

Alle Änderungen werden erst nach Neustart der Systems übernommen, welcher entweder manuell, oder im [Logout-Dialog](#) mit dem Button "Reboot" erfolgen kann.



9.3.3.3 Display

Auch das Verwaltungsfenster für die Anzeigengeschäftsstellen (Grafikkarten) ist im Aufbau und in der Funktionsweise den beiden vorhergehenden Fenstern sehr ähnlich:



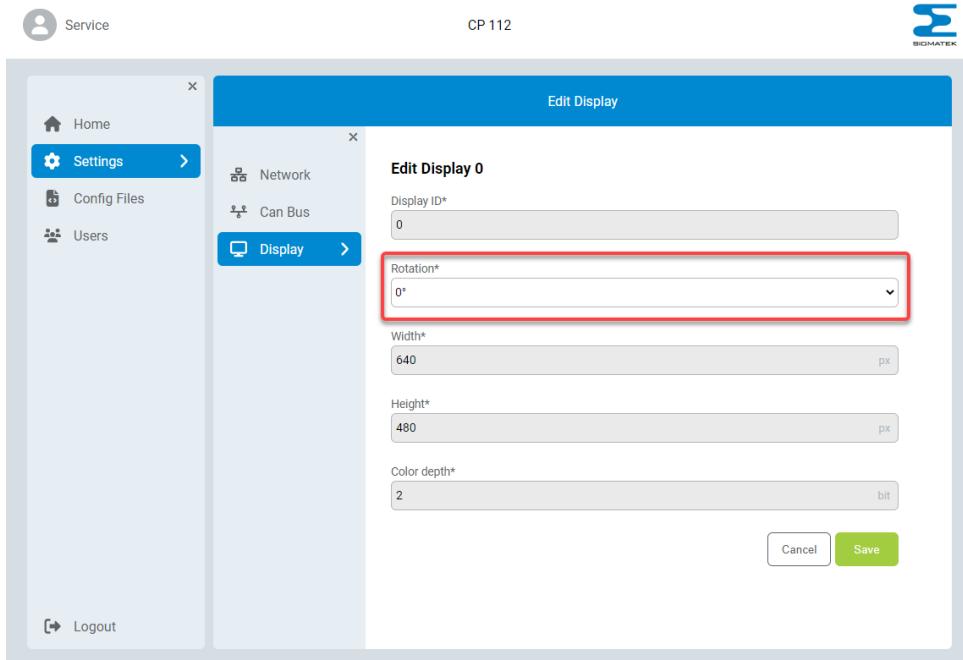
CP 112

SIGMATEK

Displays

ID	Width, Height, Color depth, Rotation	
Display 0	640px, 480px, 2bit, 0°	Edit
Display 1	800px, 600px, 4bit, 90°	Edit
Display 2	1024px, 768px, 8bit, 180°	Edit
Display 3	1280px, 1024px, 16bit, 270°	Edit

Hier fällt auf, dass alle Schnittstellen den "blauen Blitz" haben - dies ist auch einer der Hauptunterschiede zu den anderen Verwaltungsfenstern: es werden nur physisch vorhandene Anzeigen (=Grafikkarten) angezeigt. Da diese von der SIGMATEK Device Configuration selbstständig erkannt werden, kann nach dem Klick auf den Edit-Button auch nur ein Feld geändert werden, welches auch Pflichtfeld ist: die Rotation.



Service

CP 112

Edit Display

Edit Display 0

Display ID*
0

Rotation*
0°

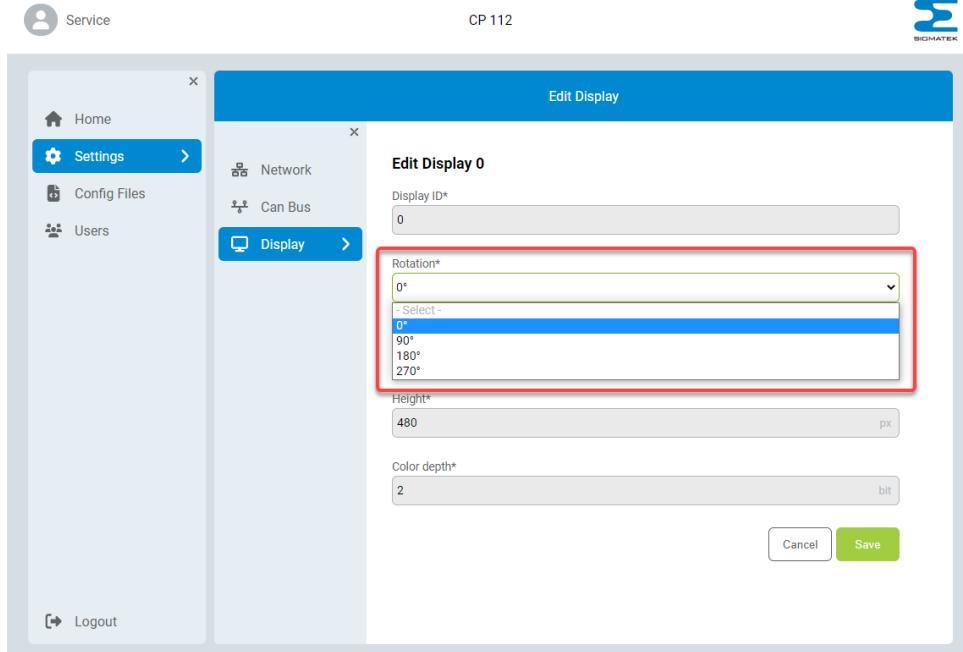
Width*
640 px

Height*
480 px

Color depth*
2 bit

Cancel Save

Hier kann zwischen den vier Drehungen jeweils um einen rechten Winkel ausgewählt werden:



Service

CP 112

SIGMATEK

Home

Settings >

Config Files

Users

Network

Can Bus

Display >

Logout

Edit Display

Edit Display 0

Display ID*

0

Rotation*

0°

0° (selected)

90°

180°

270°

Height*

480 px

Color depth*

2 bit

Cancel

Save

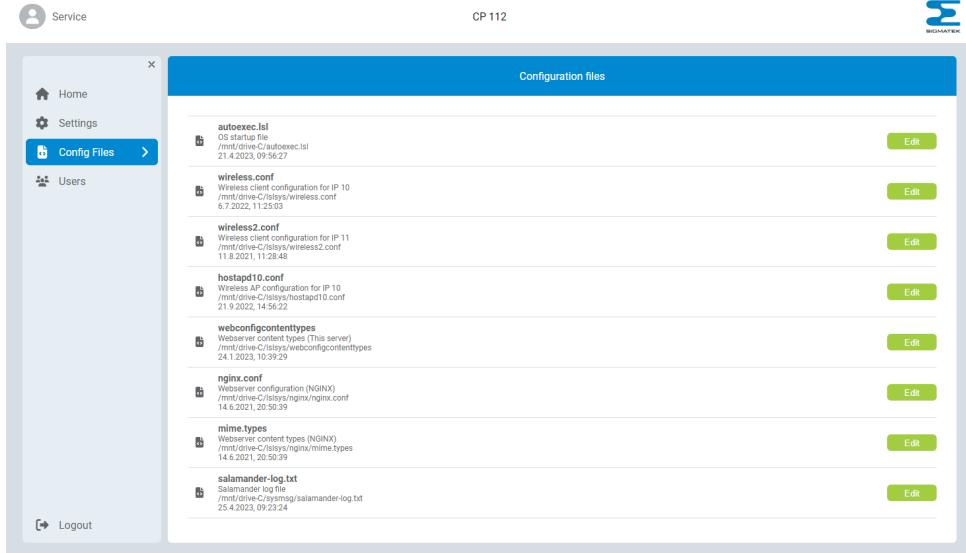
Alle anderen Felder (ID, Auflösung, Bittiefe) werden automatisch erkannt und eingetragen.

Alle Änderungen werden erst nach Neustart der Systems übernommen, welcher entweder manuell, oder im [Logout-Dialog](#) mit dem Button "Reboot" erfolgen kann.



9.3.4 Config-Dateien

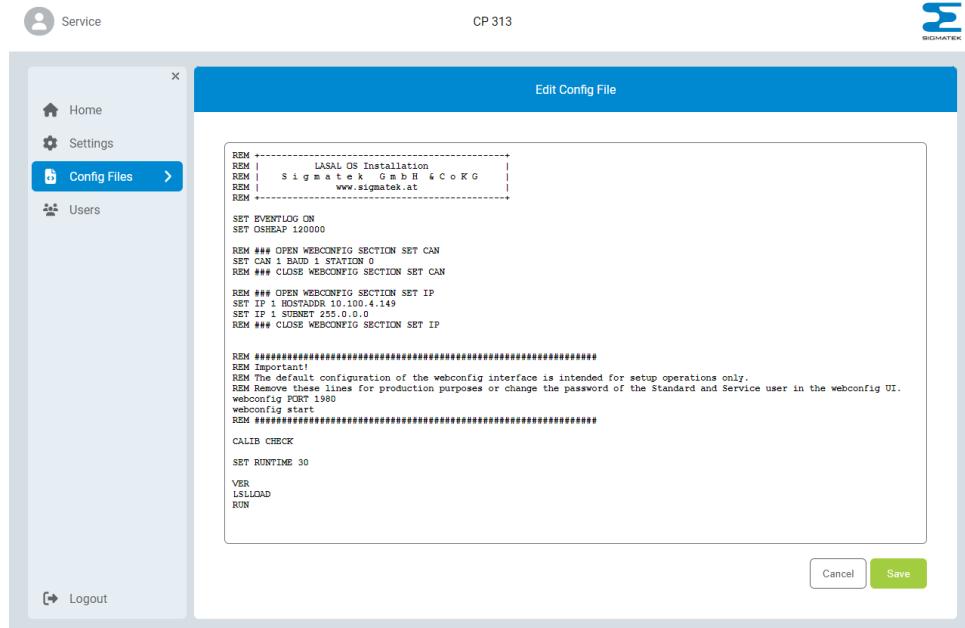
Dieser Bereich ist dem Anwender "[Service](#)" vorbehalten. Hier können die Konfigurationsdateien editiert werden. Nicht alle dieser Dateien müssen zwangsläufig vorhanden sein, manche sind optional.



The screenshot shows the 'Config Files' section of the LASAL OS interface. On the left is a sidebar with 'Home', 'Settings', 'Config Files' (which is selected and highlighted in blue), and 'Users'. On the right, a main area is titled 'Configuration files' and lists several configuration files with their details and an 'Edit' button:

File	Description	Actions
autoexec.lsl	OS startup file /mnt/drive-C/sysmsg/autoexec.lsl 21.4.2023, 09:56:27	Edit
wireless.conf	Wireless client configuration for IP 10 /mnt/drive-C/sysmsg/wireless.conf 6.7.2022, 11:25:03	Edit
wireless2.conf	Wireless client configuration for IP 11 /mnt/drive-C/sysmsg/wireless2.conf 11.8.2021, 11:28:48	Edit
hostapd10.conf	Wireless AP configuration for IP 10 /mnt/drive-C/sysmsg/hostapd10.conf 21.9.2022, 14:56:22	Edit
webconfigcontenttypes	Webserver content types (This server) /mnt/drive-C/sysmsg/webconfigcontenttypes 24.1.2023, 10:39:29	Edit
nginx.conf	Webserver configuration (NGINX) /mnt/drive-C/sysmsg/nginx/nginx.conf 14.6.2021, 20:50:39	Edit
mime.types	Webserver content types (NGINX) /mnt/drive-C/sysmsg/nginx/mime.types 14.6.2021, 20:50:39	Edit
salamander-log.txt	Salamander log file /mnt/drive-C/sysmsg/salamander-log.txt 23.4.2023, 09:23:24	Edit

Klickt man auf den Button "Edit", kann die entsprechende Konfigurationsdatei (hier als Beispiel die Datei Autoexec.lsl) geändert, und mit der "Save"-Taste gespeichert werden:



Service

CP 313

Edit Config File

```

REM
REM      LASAL OS Installation
REM      S i g m a t e k  G m b H & C o C K G
REM      www.sigmatek.at
REM

SET EVENTLOG ON
SET OSIRAP 120000

REM ### OPEN WEBCONFIG SECTION SET CAN
SET CAN 1 BASE 1 STARTED
REM ### CLOSE WEBCONFIG SECTION SET CAN

REM ### OPEN WEBCONFIG SECTION SET IP
SET IP 1 HOSTADDR 10.100.4.149
SET IP 1 SUBNET 255.0.0.0
REM ### CLOSE WEBCONFIG SECTION SET IP

REM #####
REM Important!
REM The default configuration of the webconfig interface is intended for setup operations only.
REM Do not use these lines for production purposes or change the password of the Standard and Service user in the webconfig UI.
webconfig PORT 1900
webconfig start
REM #####
REM CALIB CHECK
SET RUNTIME 30
VER
LSLLLOAD
RUN

```

Logout

Cancel Save

 Es ist äußerste Vorsicht geboten: sämtliche Änderungen an den Konfigurationsdateien werden direkt auf die Steuerung geschrieben! Beim nächsten Booten sind dann diese Werte gültig, und es kann sein, dass z.B. zu der Steuerung keine Verbindung mehr aufgebaut werden kann!

 Wird im Bereich Settings auf der Seite "Network", oder "Can Bus" eine neue Einstellung eingetragen, wird diese am Ende der Autoexec.lsl-Datei eingefügt. Will man die Position, wo die Netzwerk- bzw. Can Bus-Einstellungen eingefügt werden, innerhalb der Autoexec.lsl-Datei konfigurieren, kann der Bereich für diese Einstellungen durch spezielle Kommentare als "Section Block" eingefasst (gekennzeichnet) werden. Diese sind für die Seite "Network":

REM ### OPEN WEBCONFIG SECTION SET IP

...

REM ### CLOSE WEBCONFIG SECTION SET IP

und für die Seite "Can Bus":

REM ### OPEN WEBCONFIG SECTION SET CAN

...

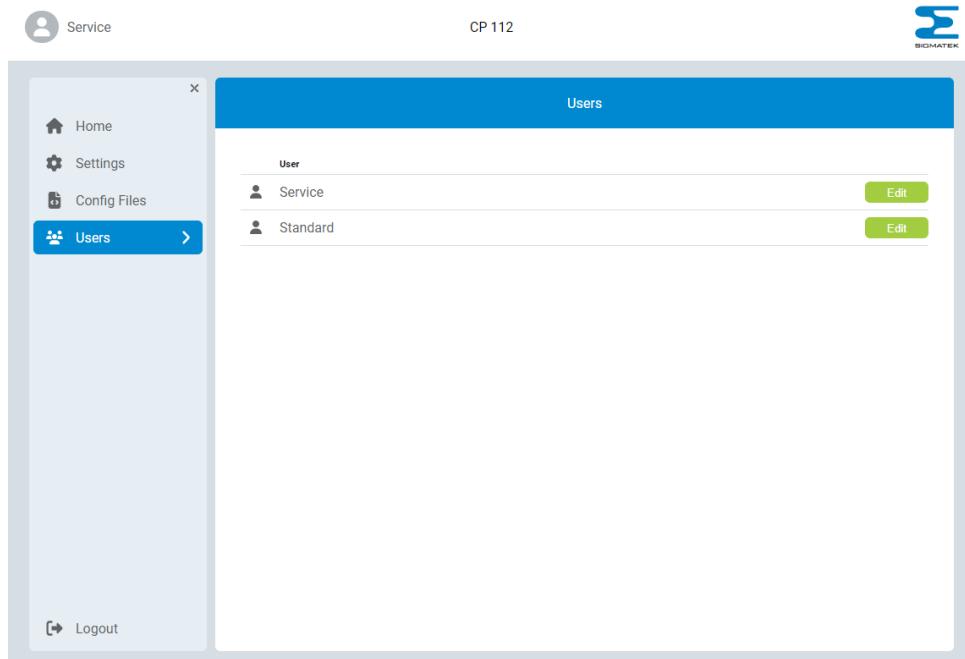
REM ### CLOSE WEBCONFIG SECTION SET CAN

Wird kein valider "Section Block" für die Einstellung gefunden, wird in einer Meldung auf der jeweiligen Seite im Settings-Bereich auf diese Tatsache sowie auf den notwendigen Syntax hingewiesen (in unserem Beispiel ist es die Seite "Network"):



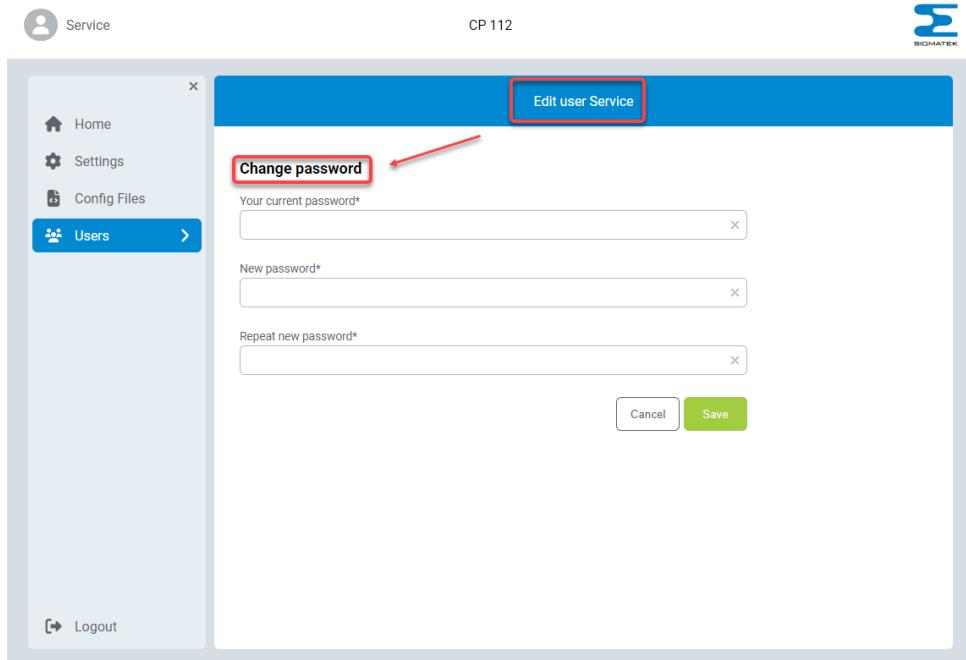
9.3.5 Users

Auch dieser Bereich ist dem Anwender "[Service](#)" vorbehalten. Hier können die Benutzer verwaltet werden:



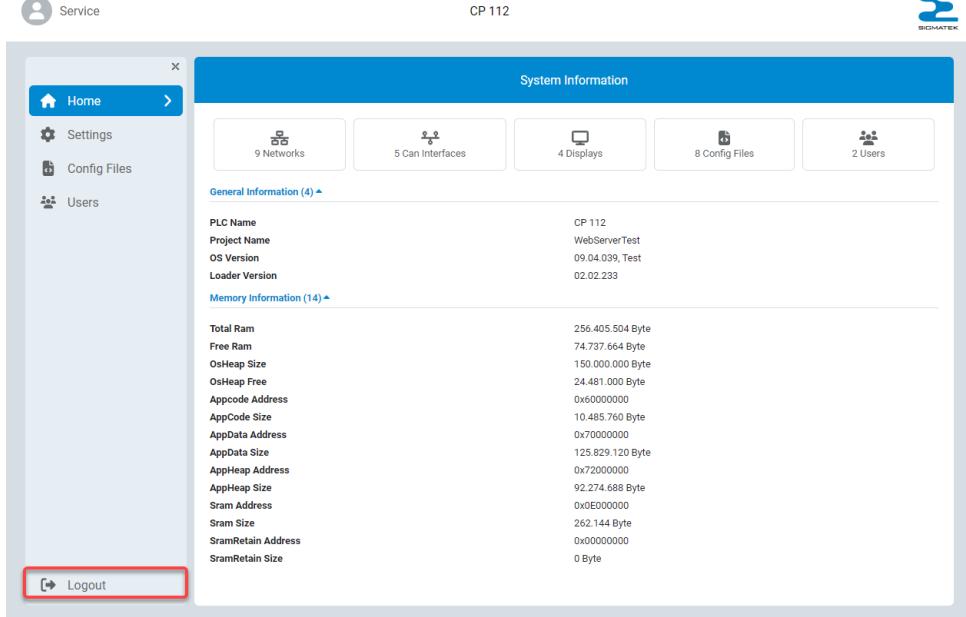
User	
Service	Edit
Standard	Edit

"Verwalten" bedeutet in diesem Fall nur, das Passwort zu ändern:



9.4 Logout

Ist die Konfiguration abgeschlossen, oder soll ein anderer Anwender angemeldet, die Webseite neu geladen bzw. die Steuerung neu gebootet werden, wird der aktuell eingeloggte User abgemeldet. Dies geschieht mit dem Logout-Button, welche sich auf der linken Seite, wo die Bereiche angezeigt werden, ganz unten (bei links stehenden Bereichsanzeigen)...



Service

CP 112

System Information

General Information (4) ▲

PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

Memory Information (14) ▲

Total Ram	256.405.504 Byte
Free Ram	74.737.654 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
AppCode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	125.829.120 Byte
AppHeap Address	0x72000000
AppHeap Size	92.274.688 Byte
Sram Address	0x0E000000
Sram Size	262.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

[Logout]

...bzw. ganz rechts (bei unten angezeigten Bereichsicons) befindet.

Service

CP 112



System Information

 9 Networks	 5 Can Interfaces	 4 Displays
 8 Config Files	 2 Users	

General Information (4) ▲

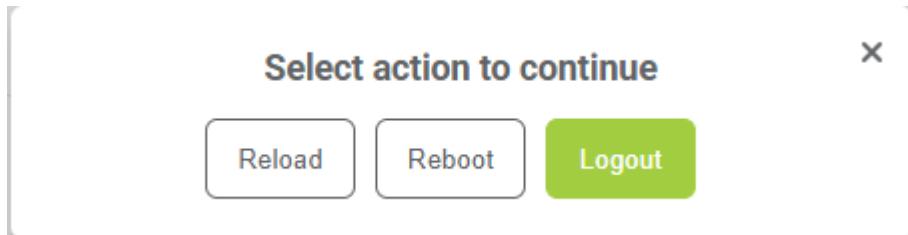
PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

Memory Information (14) ▲

Total Ram	256.405.504 Byte
Free Ram	74.737.664 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
Appcode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	125.829.120 Byte
AppHeap Address	0x72000000
AppHeap Size	92.274.688 Byte
Sram Address	0x0E000000
Sram Size	262.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

Wird auf das Logout-Symbol geklickt, erscheint folgender Dialog:



Hier hat man drei Optionen.

- Mit Reload wird die Webseite neu geladen
- Mit Reboot wird die Steuerung neu gestartet
- Mit Logout erscheint der LOGIN-Dialog und der Anwender kann sich neu anmelden



Alle Änderungen werden erst nach Neustart der Systems übernommen, welcher entweder manuell, oder hier mit dem Button "Reboot" erfolgen kann.

9.5 Benutzerverwaltung

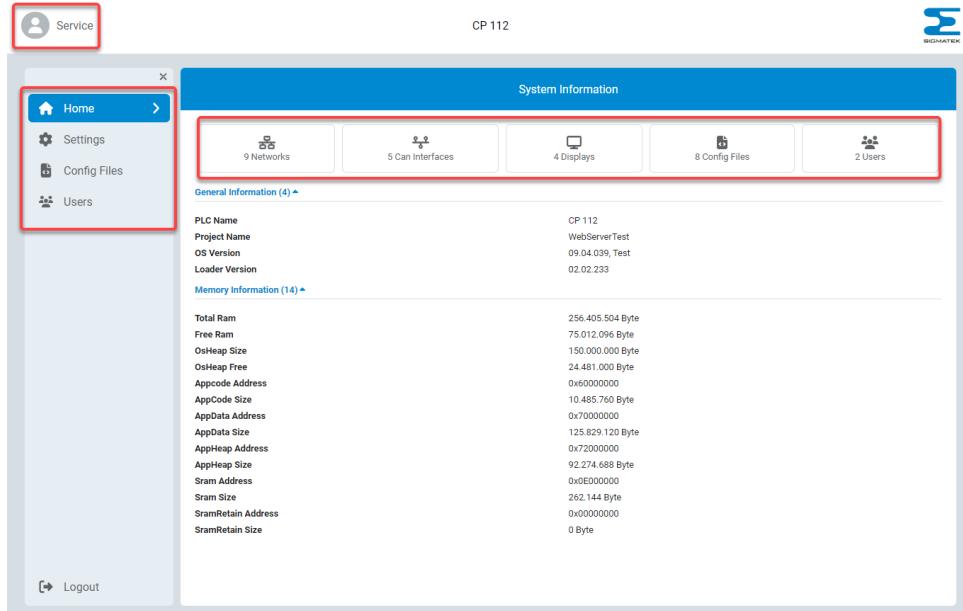
In der SIGMATEK Device Configuration sind zwei vordefinierte Anwender vorhanden: "Standard" und "Service". Was die beiden unterscheidet, und was sie können, lesen Sie in den Kapiteln

- [Der Anwender "Service"](#)
- [Der Anwender "Standard"](#)

9.5.1 Der Anwender "Service"

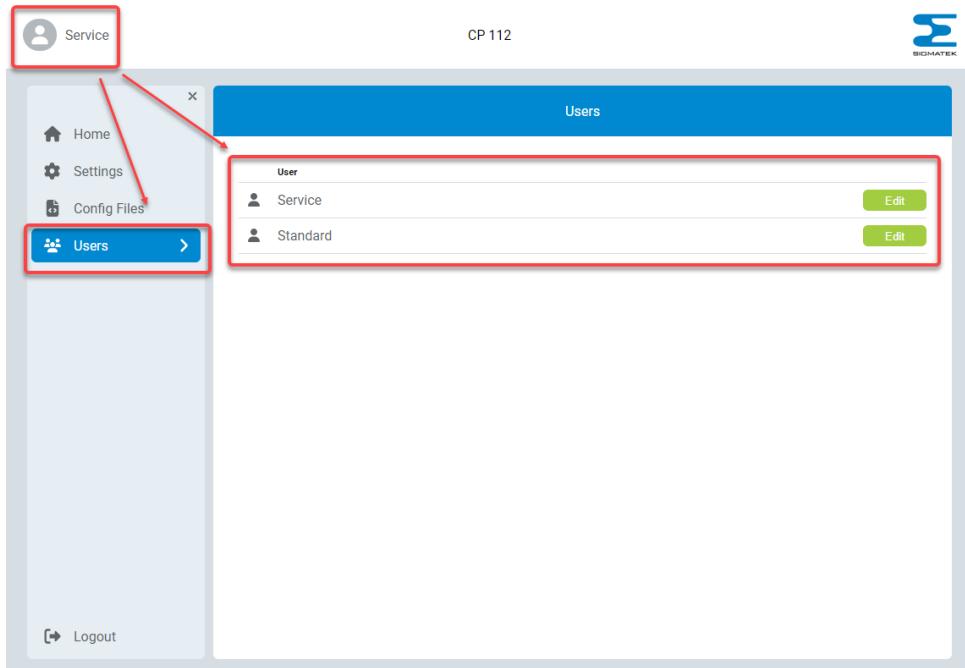
Der Anwender "Service" entspricht einem Administrator. Dieser hat alle Rechte, sprich:

- o Er kann alle vier Bereiche ([Home](#), [Settings](#), [Config Files](#), [Users](#)) sowie alle fünf möglichen Shortcuts sehen:

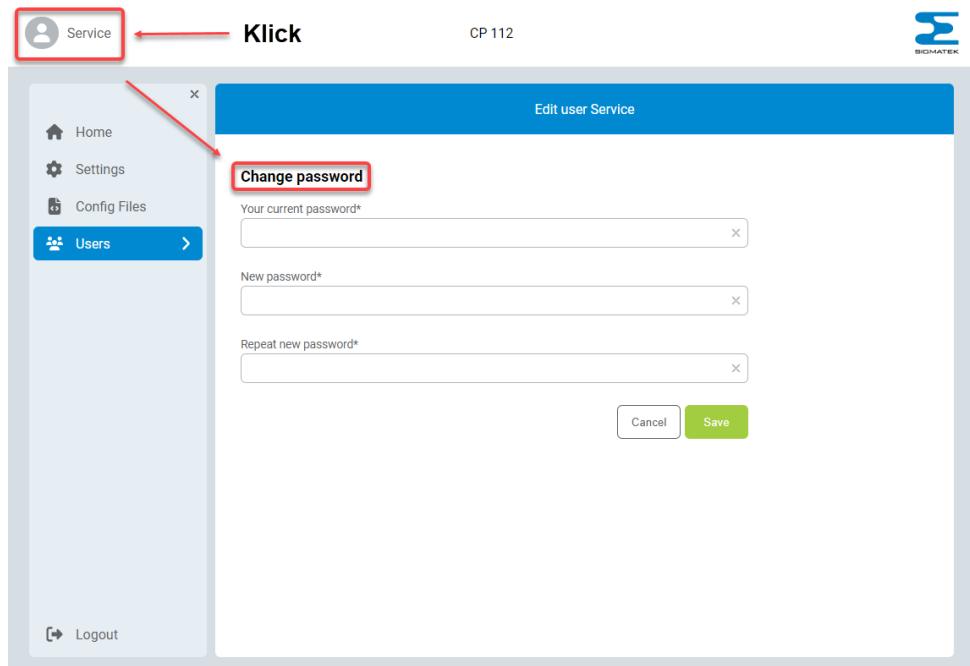


The screenshot shows the LASAL OS user interface. At the top, there is a header with the SIGMATEK logo and the text "LASAL OS". Below the header, the title "9.5.1 Der Anwender 'Service'" is displayed. The main content area shows the user "Service" logged in. The sidebar on the left contains links for "Home", "Settings", "Config Files", and "Users". The "Users" link is highlighted with a red box. The main panel displays "System Information" with icons for 9 Networks, 5 CAN Interfaces, 4 Displays, 8 Config Files, and 2 Users. Below this, there are two sections: "General Information (4)" and "Memory Information (14)". The "General Information" section lists PLC Name (CP 112), Project Name (WebServerTest), OS Version (09.04.039, Test), and Loader Version (02.02.233). The "Memory Information" section lists various memory addresses and sizes, such as Total Ram (256.405.504 Byte), Free Ram (75.012.096 Byte), and AppCode Address (0x60000000). At the bottom of the main panel, there is a "Logout" button.

- o Er kann nicht nur das eigene, sondern auch das "Standard"-Konto verwalten, sprich: das Passwort ändern, indem auf dem Bereich "[Users](#)" geklickt wird:



Wird hier beim Anwender "Service" auf "Edit" geklickt, entspricht das dem Klick auf den Benutzernamen links oben:



CP 112

Service

Klick

Edit user Service

Change password

Your current password*

New password*

Repeat new password*

Cancel Save

Home

Settings

Config Files

Users

Logout

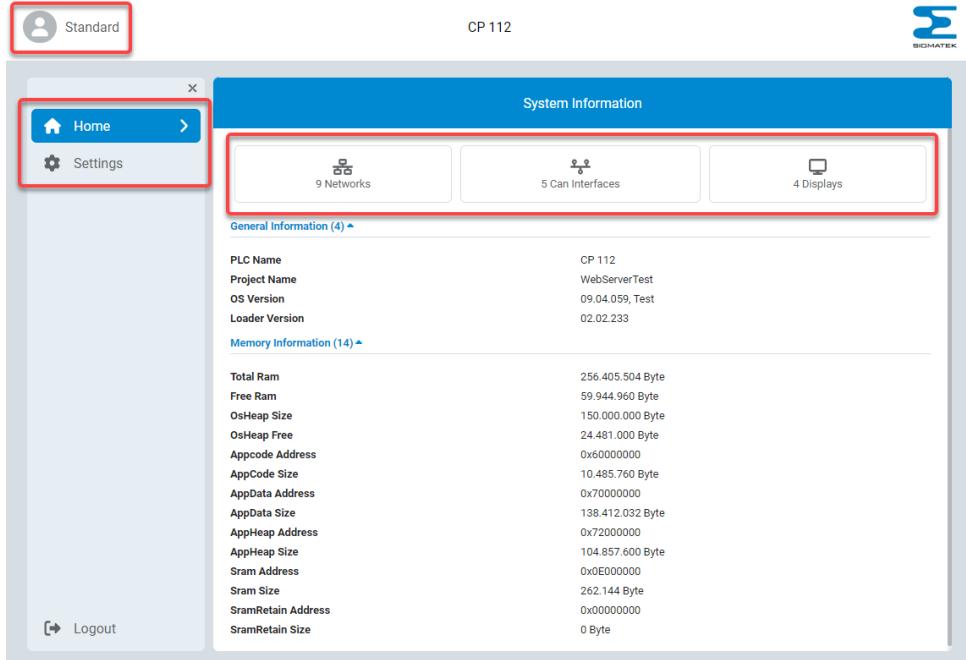


Das Passwort für den User "Service" ist im Auslieferungszustand "sigmatek".

9.5.2 Der Anwender "Standard"

Der Anwender "Standard" hat nur eingeschränkte Rechte, sprich:

- Er kann nur die ersten beiden Bereiche ([Home](#), [Settings](#)) sowie nur drei aus den fünf möglichen Shortcuts sehen:



CP 112

System Information

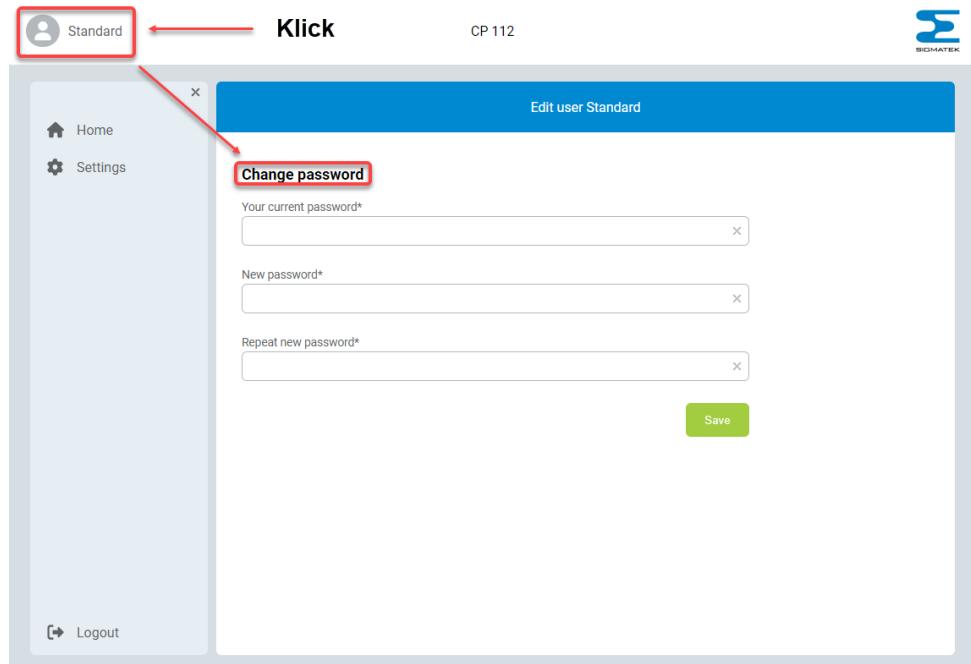
General Information (4) ▾

PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.059, Test
Loader Version	02.02.233

Memory Information (14) ▾

Total Ram	256.405.504 Byte
Free Ram	59.944.960 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
Appcode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	138.412.032 Byte
AppHeap Address	0x72000000
AppHeap Size	104.857.600 Byte
Sram Address	0x0E000000
Sram Size	262.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

- Er kann nur das eigene Konto verwalten, sprich: das eigene Passwort ändern:

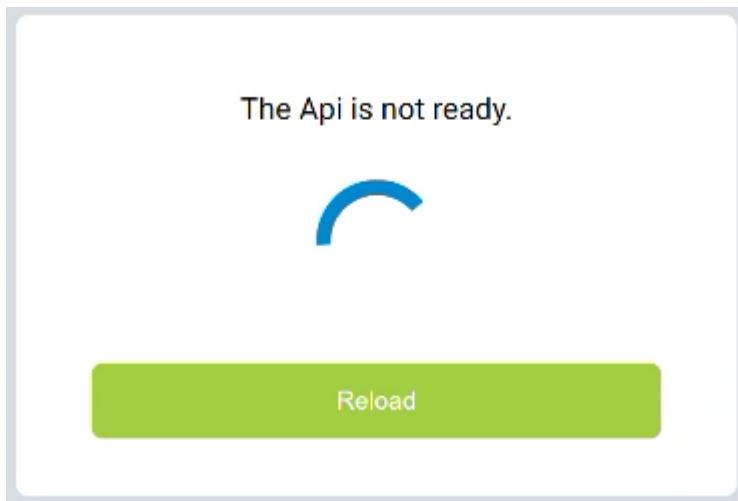


Das Passwort für den User "Standard" ist im Auslieferungszustand "sigmatek".



9.6 Troubleshooting

Um mit SIGMATEK Device Configuration eine Steuerung zu konfigurieren, ist es erforderlich, dass neben einer funktionierenden Verbindung zu der Steuerung auch eine funktionierende API im Hintergrund vorhanden ist. Wenn letztere aus irgendeinem Grund temporär nicht erreichbar ist, wird eine entsprechende Meldung angezeigt:



Diese Anzeige ist so lange sichtbar, bis die API wieder zur Verfügung steht. Ist dies der Fall, so wird ein [LOGIN-Fenster](#) angezeigt, und der Anwender wird aufgefordert, sich vom Neuen anzumelden.

9.7 Disclaimer

Produkte, welche im Auslieferungszustand die Betriebssystem- Versionen

- Salamander 2 ARM + Salamander 2 x86 \geq Version 09.02.200
- Salamander 3 \geq Version 09.04.060 bzw.
- Gecko ARM + Gecko x86 \geq Version 09.07.100

- Salamander 2 ARM + Salamander 2 x86 \geq Version
09.02.200
- Salamander 3 \geq Version
09.04.060 bzw.
- Gecko ARM + Gecko x86 \geq Version
09.07.100
 - vor der Auslieferung der Steuerung an einen Endkunden
 - vor einer anderweitigen Verwendung im Produktivbetrieb (sowohl hausintern als auch bei Endkunden oder Dritten)
 - vor der Verbindung des Geräts mit einer Netzwerkinfrastruktur, bei welcher eine Zugriffsmöglichkeit durch Dritte von außen gegeben ist