

LASAL

Operating System

Operating Manual

Publisher: 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

Translation from German

All rights reserved. No part of this work may be reproduced, edited using an electronic system, duplicated or distributed in any form (print, photocopy, microfilm or in any other process) without the express permission.

We reserve the right to make changes in the content without notice. The SIGMATEK GmbH & Co KG is not responsible for technical or printing errors in the handbook and assumes no responsibility for damages that occur through use of this handbook.

Table of Contents

LASAL OS	37
User Guide	38
User Guide	38
Software Structure of a Lasal CPU	38
The Loader	40
Comlink Interface	43
Updating the Operating System	46
Backup of Persistent Data	46
Drive Letter Assignments	46
Logging of System and Application Messages	50
Operating System Messages	50
Runtime Check and Watchdog	57
Hardware Watchdogs	58
Software Checks	58
Error Detection	59
Flow of Operation	61
CIPC Peripheral Reset	62
DIAS Slave Peripheral Reset	63
Simple Router	64
Security Downgrade	65
LASALOS Supported USB Devices	65
LASALOS Supported USB Devices for CCL722, DCL642 and DTC081	66
USB Update	67
How to Setup/Defragment a Compact Flash Medium	70
Lasal OS Tasks	70
Assignment of CAN Identifiers	72
PLC (online + comlink protocol)	72
MiniDisplay	73
Enumerating Ethernet Interface Connections	74
Status and Error Messages	74
Routing	81

Overview	81
Commands	82
Routing Table Syntax	82
Boot Screen	83
File System for USB Sticks	84
Task Management	84
Threads	84
LASAL Tasks	85
Programming Guide	87
The Construct	87
Exception Handling	88
APIs	89
OS_SSR_SetHandler	89
OS_SSR_SkipOverDIV	90
Command Line Interface CLI	92
Overview	93
File System Commands	93
Operating System Commands	93
LASAL Data File (active.dat) Commands	94
Kernel Error Log Commands	94
Projects Commands	94
Configuration Commands	94
Configuration Commands (IP)	94
Configuration Commands (CAN)	95
Configuration Commands (DIAS)	95
Configuration Commands (SERIAL)	95
Configuration Commands (Mouse/Touch)	95
Configuration Commands (Printer)	95
Batch Processing Commands	95
Environment Variables	95
Command Line Syntax	96
Keys and Files	96
Order of Execution of Commands in autoexec.lsl	96

File System Commands	97
ARCH	97
ATTRIB / ATTR	98
CD / CHDIR	99
CHKDSK	100
COPY / C	101
CREATE	101
DEL / ERASE	102
DIR / D	103
DRIVES	104
FC	105
FORMAT	105
LABEL	105
MD / MKDIR	106
MOUNT	106
PARTITION	107
RD / RMDIR	108
REN	108
SMBSVR	109
TREE	110
TYPE	110
UMOUNT	111
UNZIP	111
VOL	112
XCOPY	112
ZIP	113
Operating System Commands	113
ADDR	113
APPTASKS	114
ARP	114
BGTASKS	114
BIOS	114

BOOT	114
BOOTPIC	117
BOXES	117
BUFFERS / BUF / B	117
CANINFO	118
CIL	118
CYTASKS	118
DATE	118
DUMP	118
ERRORLOG	119
EXCEPT	119
FCLOSEALL	120
FILE	120
FILES	121
FLUSHLOG	121
GET ERRORLOG	121
HANDLES	122
HEAP	122
HELP / ?	123
INFO	123
INTS	124
LINK	125
LISTTS	125
LOG	126
MEM	126
MEMDUMP	126
NETSTAT	127
NETWORK	127
NUMLOCK	127
PACKAGE	128
PING	129
PNPBIOS	129

QUCMD	129
REBOOT	130
REXX	130
ROUTE	130
RTTASKS	131
SEMAS	131
SET CYCLIC	131
SET DBGHEAP	132
SET DEBUG	134
SET ERRORLOG	134
SET EVENTLOG	134
SET SYSTRACE	135
SET TRACEBUFSIZE	137
SET USB_CACHE_FLUSHING	137
SETENV	138
SETLEDSTATUS	139
SHOWENV	139
SRAMLOAD	140
SRAMSAVE	140
STATUS	140
TASKS1 / TASKS	141
TASKS2	141
UNMOUNT	141
VER	141
vpndaemon	142
VT	143
Projects Commands	145
SET FPUROUNDCONTROL	145
LSLLOAD / LO	146
LSLSAVE / SA	146
RUN	147
RESET	147

PROJECT	147
Configuration Commands	148
ADDDBR	148
CALIB	149
Calib check	149
CHANGECERT	150
DIAS	150
EURO	151
FIREWALL	151
IP	154
IPCINFO	155
SCREENSAVER	155
SET	156
SET APPCODE	156
SET APPDATA	157
SET APPLSAVE	157
SET BOOTTEXT	158
SET BTRUNTIME	158
SET CAN	158
SET CLI	160
SET CPUBT	160
SET CPUTC	160
SET CPURT	161
SET DATE	161
SET DBGLEVEL	161
SET DIASERROR	166
SET DISPLAY	166
SET DISPLAYRESOLUTION	167
SET DISPLAYROTATE XX	168
SET FIRSTUSBDRIVE	168
SET FORCEXTSINGLE	169
SET IP	169

SET IP PORT	171
SET KEYS	171
SET LASALCOM	172
SET LCD	173
SET MAINTIMER	173
SET MOUSE	174
SET MULTICOREOJBS	175
SET MULTI_VM	176
SET OSHEAP	176
SET OSZIMEM	177
SET PRINTER	177
SET RAM	178
SET ROUTINGTABLE	178
SET RTRUNTIME	179
SET RUNTIME	179
SET SECURE	179
SET SRAMRETAIN	180
SET STATION	180
SET STOPWAIT_TIME	181
SET TIME	181
SET VISU	181
SET WATCHDOG	182
TOUCHCFG	182
TOUCHTEST	182
UPDATETOUCH	182
US	183
Batch Processing Commands	183
CLS	183
DELAY	183
ECHO / @ECHO	184
EXIT	184
GOTO	184

IF EXISTS	185
PAUSE	185
REM	185
Enviroment Variables	186
FILENAMES	186
REBOOTONPF	186
LasalOS File Error Return Codes	186
Software Configuration	189
Software Components	189
Installation with the Debug Tools LASAL Class and LSE	190
Installation via the Command Line Interface (CLI)	191
Automatic Installation of a Exchangeable Storage Media	191
Rexx Programs for Installing Software	192
Description File of a Software Package	193
Examples for Description Files	194
Program to Create an Archive File	195
AUTOSTRT REXX Program for installation software package	196
Example	196
System Update IPC	196
System Update CPU	197
LASAL on Several Disks	198
Rexx Interpreter	200
What is LasalOS Rexx?	201
The Purpose of this Document	201
LasalOS-specific Built-in Functions	203
DIRLIST	204
DRIVEINFO	205
FILEINFO	206
GETCH	207
GETCHE	208
GETCPUSTATUS	208
GETDATAD	208

GETOSVERSION	209
GETOSVERSION2	209
GETRES	210
HWINFO	211
HWINFOONLINE	212
KBHIT	213
LSLLINK	213
LSLLOADPRJ	213
LSLSEND	214
LSLSENDMOD	214
LSLSENDMODCLSNAMEx	214
MILLISLEEP	215
OFFLINE	215
ONLINE	215
RESETSPS	216
REXEC	217
RUNSPS	217
RXFILE	217
SENDOS	218
SERVERREAD	218
SERVERWRITE	219
SETBREAKPOINT	219
TXFILE	220
Command Line Interface (CLI) Commands	221
Appendix A – Sample Rexx programs	221
Appendix B – Error Codes	224
Appendix C – REXX How To	225
Function Call	225
Conditions and Loops	227
Online Functions	227
ONLINE	228
OFFLINE	229

Example Function	229
Executing CLI Commands	230
Address System cmd	230
REXEC	231
SHOWENV	231
Pipe Operator „>“	232
GETDATAD	232
File System Operations	233
stream	233
lineout	234
linein	235
lines	235
attrib	236
REXX Error Codes	237
-307 – Negative command received	237
Useful Code Sections	237
Converting	237
Reading and Processing	237
Creating Log File	239
Creating a Path	241
Copying a Folder 1	244
Copying a Folder 2	246
Copying a Folder 3	249
Notepad++ Highlighting	254
Links	257
Code Highlighting	257
VNCcli	265

What is LasalOS VNCcli?	265
The Purpose of this Document	265
LASAL OS VNCcli-API Functions	266
The OS Interface VNCCLI	266
Callback Function	267
udSize	270
udVersion	270
udVersionVNC	270
VNCHandle	271
VNCConnect	272
VNCDisconnect	273
VNCDisconnectHandle	274
VNCDraw	275
VNCDraw2	275
VNCExit	276
VNCForceRepaint	277
VNCFreeBuffer	277
VNCFreeOldSockets	278
VNCGetSize	278
VNCReset	279
VNCSendHIDEEvent	279
VNCSendKbrdEvent	280
VNCSendKeyboardEvents	281
VNCSendPointerEvents	282
VNCSetCliSleep	282
VNCSetConnectTimeout	283
VNCSetKaValues	284
VNCSetOptions	285
VNCSetTaskDelayTime	287
Error messages in Event00.log file	287
Error Codes VNC Client	289
Command Line Interface (CLI) Commands	290
VNCsvr	290

What is LasalOS VNCSvr?	290
The Purpose of this Document	291
LASAL OS VNCSvr-API Functions	292
The OS Interface VNCSVR	292
udSize	293
udVersion	293
udVersionVNC	294
VNCSVRAcceptKeyboardEvents	294
VNCSVRAcceptPointerEvents	295
VNCSVRAlwaysShared	295
VNCSVRClientWaitTimeMillis	296
VNCSVRCompareFB	297
VNCSVRDisconnectAllClients	297
VNCSVRDisconnectClient	298
VNCSVRConnectedClients	298
VNCSVRConnectedClients2	299
VNCSVRDisconnectClientsOnNonsharedConnection	299
VNCSVRDisableSharedMemory	300
VNCSVREnumConnections	301
VNCSVREnumConnections2	301
VNCSVRExit	302
VNCSVRFullRefreshCycle	303
VNCSVRInit	303
VNCSVRInitialized	305
VNCSVRMaxConnections	305
VNCSVRNeverShared	306
VNCSVRReset	306
VNCSVRSetDisplayOffset	307
VNCSVRSetFilter	307
VNCSVRSetPass	308
VNCSVRTimeoutInactiveClient	309
VNCSVRUpdateRect	310

Error Codes VNC Server	310
Command Line Interface Commands	311
Configure the VNC Server Using Environment Variables	311
VNC_SVR_ACCEPT_KEYBOARD	311
VNC_SVR_ACCEPT_POINTER	312
VNC_SVR_ALWAYS_SHARED	312
VNC_SVR_BLACKLIST_LEVEL	312
VNC_SVR_COMPARE_FB	313
VNC_SVR_DESKTOP_NAME	313
VNC_SVR_DISCONNECT_ON_NONSHARED	313
VNC_SVR_DISABLE_SHARED_MEMORY	314
VNC_SVR_FILTER	314
VNC_SVR_FORCE_BPP	314
VNC_SVR_FULL_UPDATE	315
VNC_SVR_LOGFILE	315
VNC_SVR_LOGSTR	315
VNC_SVR_MAX_CONNECTIONS	316
VNC_SVR_NEVER_SHARED	316
VNC_SVR_PORT	316
VNC_SVR_PORT2	317
VNC_SVR_TIMEOUT_INACTIVE	317
VNC_SVR_WAIT_FOR_CLIENT	317
Repeater Function	317
Starting the Repeater Function	318
Using the Repeater Function	319
Configuring the Repeater Function	320
Sample Configurations Using the AUTOEXEC.LSL	320
Scenario 1: The server should allow exactly one connection from a specific host.	320
Scenario 2: The server should allow every host to connect, but only for viewing.	321
Scenario 3: Setting up the server using a separate log file.	321
Scenario 4: Set up a server which will write remote events into the user-log event02.log.	322

Example Configurations with the vncsvr.cfg File	322
Scenario 1: VNC server logs in with the serial number	322
Scenario 2: VNC server logs in with an individual ID	322
Scenario 3: VNC-Server logs in with DNS	322
SRAM	323
Formats	323
Format 1	323
Format 2	323
Converting between Formats	324
Retentive Servers	324
RamEx	324
Save, Restore, Delete	325
Backing up the SRAM	325
Deleting the SRAM	325
Loading a Reorganisation Copy	325
LASAL CLASS Tool Ram Image	326
SRAMDisk	326
SRAMDisk Backup Mechanisms	327
Backing Up Data during Operation and when Switching Off	328
Backing Up Data Only When Switching Off	328
Advantages and Disadvantages of these Two Ways of Backing Up	329
Monitoring the Number of Writing Operations to the SD Card	329
Detecting an Error when Writing to the SRAMDisk during Power-down	330
Response when an Error is Detected in the SRAM	330
Reorganizing the SRAM	330
SRAM Size and Memory Allocation	331
Repeater	332
LASAL OS with Repeater Function	332
Configuration	333
lslrepeater.cfg	333
Command	334
Example Configuration with the lslrepeater.cfg File	334

Scenario 1: Control Logs in with Serial Number	334
Scenario 2: Control Logs in with Individual ID	334
Scenario 3: Control Logs in via DNS	335
LASAL Tools	335
LASAL Repeater	336
Remote Diagnosis	336
Operating System Interfaces	339
Variables	339
Content of the OPS Structure	342
Content of the _OnlineMap Structure	344
API MultiTask	344
General Instructions	344
Multitasking	344
Time	350
Message Passing	351
Semaphore Handling	351
Mailbox Handling	353
Multitask Class: Interface Functions	354
Functions	354
Extended Functions	355
GETLASTERROR	355
GETTIME	356
Task Functions	356
CREATETHREAD	356
CURRENTTASKHANDLE	358
DELAYUNTIL	359
GETMINSTACK	359
GETTASKPRIORITY	360
GETTASKSTACK	360
GETTASKSTATE	361

RECEIVE	362
RECEIVECOND	362
RECEIVETIMED	363
RESUME	364
SEND	364
SENDCOND	365
SENDTIMED	365
SETPRIORITY	366
SUSPEND	366
TASKDELAY	367
TERMINATETASK	367
 Semaphore Functions	368
CREATESEMAPHORE	368
DELETESEMAPHORE	370
PULSE	370
RESETEVENT	371
RESOURCEOWNER	371
SEMAVALUE	372
SIGNAL	372
WAIT	373
WAITCOND	374
WAITTIMED	374
 Mailbox Functions	375
CLEARMAILBOX	375
CREATEMAILBOX	376
DELETEMAILBOX	376
GET	377
GETCOND	377
GETTIMED	378

MESSAGES	379
NEXTCOND	379
PUT	380
PUTCOND	380
PUTFRONT	381
PUTFRONTCOND	381
PUTFRONTTIMED	382
PUTTIMED	382
Interface Functions	383
Examples	384
API Application Heap	387
Overview	387
Using the Debug Heap	387
SSR Interface Functions for the Heap	388
OS_SSR_Malloc	389
OS_SSR_ReAlloc	389
OS_SSR_Free	390
API Serial Interface	390
Overview	390
Interface Functions SERIAL	391
SERUSER_ClearRecvBuffer	391
SERUSER_Close	391
SERUSER_EnableFIFO	392
SERUSER_Get422Mode	393
SERUSER_GetError	394
SERUSER_GetInfo	395
SERUSER_GetModemControl	397
SERUSER_GetModemStatus	397
SERUSER_GetRecvStatus	398
SERUSER_GetSendStatus	399
SERUSER_Init	399
SERUSER_RecvBlock	401
SERUSER_RecvChar	402

SERUSER_Send	402
SERUSER_Set422Mode	403
SERUSER_SetBufferRecv	404
SERUSER_SetModemControl	405
SERUSER_SetOnline	406
SERUSER_UserFunction	407
API Serial Number	408
General	408
Structures Used in ISYSSERNUM	408
tagPLCInfo	408
Interface Functions ISYSSERNUM	409
SernumGetPLC	410
SernumGetPLCDrive	411
SernumGetPLCInfo	412
API Sysmsg Interface	412
Overview	412
Interface Functions SYSMSG	414
OS_SYSMSG_LCLOSE	414
OS_SYSMSG_LCREATE	414
OS_SYSMSG_LFLUSH	415
OS_SYSMSG_LINFO	416
OS_SYSMSG_LOPEN	417
OS_SYSMSG_LPRINTFLN1-4	417
OS_SYSMSG_LWRITE	418
OS_SYSMSG_WRITE_I	419
OS_SYSMSG_LWRITELN	420
Sysmsg – Changes in Newer OS	421
Extended Userlog Functions	422
OS_USERLOG_EXPORT	422
OS_USERLOG_FILEHEADER	422
OS_USERLOG_FLUSH	423
OS_USERLOG_LAST_EXPORT_RESULT	424
OS_USERLOG_PRINTFLN (1-2)	425

OS_USERLOG_WRITEEVENT	426
API TCP User Interface	427
Error Values	427
How to Use OS Functions	430
OS TCP USER FUNCTIONS	431
OS_TCP_USER_ACCEPT	431
OS_TCP_USER_CLOSESOCKET	433
OS_TCP_USER_CONNECT	434
OS_TCP_USER_GETCOMLINKPORT	435
OS_TCP_USER_GETERRORSTRING	435
OS_TCP_USER_GETLINKSTATUS	436
OS_TCP_USER_GETPEERIP	437
OS_TCP_USER_GETPEERPORT	438
OS_TCP_USER_GETSERVBYNAME	439
OS_TCP_USER_GETSOCKIP	440
OS_TCP_USER_GETSOCKPORT	441
OS_TCP_USER_IOCTLSOCKET	442
OS_TCP_USER_IPINFO	443
OS_TCP_USER_LISTEN	445
OS_TCP_USER_NREAD_AVAILABLE	446
OS_TCP_USER_PING	447
OS_TCP_USER_RECV	448
OS_TCP_USER_RECVFROM	450
OS_TCP_USER_SELECT	452
OS_TCP_USER_SEND	453
OS_TCP_USER_SENDTO	455
OS_TCP_USER_SETSOCKOPT	457
OS_TCP_USER_SHUTDOWN	463
OS_TCP_USER_SOCKET	464
OS_TCP_USER_SOCKET_EX	465
OS_TCP_USER_STRTOULONG	465
OS_TCP_USER_STRTOULONG_ASY	466
OS_TCP_USER_TOIP	467

OS_TCP_USER_ULONGTOSTR	468
OS_UDP_USER_BIND	469
OS_UDP_USER_SOCKET	471
Defines	472
#define IP_ADDR_ANY	472
#define IP_ADDR_BROADCAST	472
#define IP_OPT_ADDR	472
#define IP_OPT_ETHERNET_ADDR	472
#define IP_OPT_SUBNETMASK	472
Enumeration of Ethernet Interfaces	472
Examples	473
API XTimer	476
General	476
Interface Functions XTIMER	476
XtimerInit	476
XTimerSetInterval	478
XTimerSetMode	479
XTimerStopAllUser	480
XTimerValue	481
XtimerReset	482
XtimerStop	482
XTimerStart	483
Web Server	484
Instruction	484
Requirements	484
Installation	484
Security	485
IP Address Authentication	485
Authentication with User Name and Password	486
Additional Functions	487
Command Line Interface (CLI) Commands	487
WEBSVR, WEBSVR INFO	487
WEBSVR START	488

WEBSVR AUTH	488
WEBSVR FILTER	492
WEBSVR STOP	494
Environment Variables	494
WEBROOT	494
WEBMAXAUTH	494
WEBDEFPWD	495
Example Configuration	495
LASAL OS Web Server API	496
OS_WEB_FindStringInBuffer	497
OS_WEB_GetErrorStringByValue	498
OS_WEB_GetLineFromBrowser	498
OS_WEB_InetStrToByte	500
OS_WEB_RegisterGetCallback	501
OS_WEB_RegisterPostCallback	503
OS_WEB_SendLineToBrowser	506
OS_WEB_SetAuthentication	507
OS_WEB_SetBrowserList	509
Quick Review	510
Error Codes	511
API Example	511
HTML Source Code: index.htm	512
HTML Source Code: ApplCGI.htm	512
HTML Source Code: Err401.htm	513
LASAL Class Project WebServer	514
LASAL CLASS Project WebServer Structure Text source code	514
Autoexec.lsl settings	520
Error, Warning and Info logging	520
Appendix	521
Types	521
Error Values	521
Flags	522
OS Interface Library Classes	523

Memory Library Classes	523
RamFile	523
Interfaces	523
Global Methods	524
GetDataAt	524
GetDataAtBackground	524
GetFileState	525
GetUserVersion	525
SetDataAtBackground	525
SetDataAt	526
SetSize	526
SetSizeBackground	527
SetUserVersion	527
Checksum Calculation	527
Encoding	527
RamFileRingBuffer	527
Interfaces	528
Global Methods	529
GetDataRB	529
GetDataRBBackground	529
GetLastDataRB	530
GetLastDataRBBackground	530
GetNumberOfValidEntries	530
RecordDataRB	531
RecordDataRBBackground	531
RecordRingbufferSize	531
RecordRingbufferSizeBackground	532

RecordUserVersion	532
Checksum Calculation	533
Encoding	533
LARS	534
LARS	534
Introduction	534
System Requirements	534
Installation	534
Configuration	534
LARSConfigTool	535
Settings Wizard	536
Path Setting	537
Project Setting	539
Window Setting	541
Expert Settings	542
Reset Window Position	542
Edit Autoexec.lsl	542
Ping screen project server stations	542
LARS Memory and Port Setting	543
LARS Drive Mapping	544
Repeater connection configuration	545
Using LARS	546
Communication Interfaces	546
Programming Guidelines	546
Printing with LARS	548
Communication and Driver	549
Online Communication	549
DebugIP Communication	549
Comlink	550
Online Communication	553
Communication Driver Lasal32 API	553
Overview	553
Lasal32 API for Windows	553

Lasal32 API for Linux	553
Lasal32 API Functions	553
Error Handling	554
Managing the Online Connection	556
IsOnline	556
IsOnlineH	556
LslPing	556
OcbClose	557
OcbOpen	558
Offline	558
OfflineH	558
Online	558
OnlineH	562
OnlineOptions	562
OnlineOptionsH	562
OnlinePwd	563
OnlinePwdH	564
OnlineSSL	564
OnlineSSLH	565
Modem Functions	567
ModemOpen	567
ModemGetNumber	567
ModemClose	567
ModemCall	570
ModemGetName	571
Data Exchange Functions	571
GetData	572
GetDataH	573
GetDataList	573
GetDataListH	574
LslExecKillH	575
LslExecNewInstrEx	577
LslExecNewInstrExH	577

LslGetAdressVar	579
LslGetAdressVarH	579
LslGetDataEx	580
LslGetDataExH	580
LslGetDataListEx	581
LslGetDataListExH	581
LslGetDataListSpecial	582
LslGetDataListSpecialH	583
LslSetDataEx	584
LslSetDataExH	584
SendDataList	585
SendDataListH	585
SetData	587
SetDataH	587
Status Information and PLC Commands	588
GetBusType	588
GetBusTypeH	588
GetChk	589
GetChkByType	589
GetChkByTypeH	590
GetChkH	591
GetCpuStatus	592
GetCpuStatus2	592
GetCpuStatus2H	592
GetCpuStatusH	593
GetPowerFailInfo	594
LslGetDIIIInfo	595
LslGetDIIIVersion	595
LslGetMemInfo	596
LslGetMemInfoH	596
LslGetPLCInfo	598
LslGetPLCInfoH	599

LslGetProjectInfo	601
LslGetProjectInfoEx	601
LslGetProjectInfoExH	601
LslGetProjectInfoH	603
LslReadDateTime	604
LslReadDateTimeH	605
LslSetDateTime	605
LslSetDateTimeH	605
LslSyssernumCommand	606
LslSyssernumCommandH	606
SetCommand	608
SetCommandH	608
Administrative Functions	608
SetMultiThreadSupport	609
File Functions	609
CB_FORMAT_FUNCTYPE	609
FileInfo	611
FileInfoH	611
FileLoad	613
FileLoadH	613
FileSave	615
FileSaveEx	615
FileSaveExH	615
FileSaveH	618
FileSetAttributes	621
FileSetAttributesH	621
LslCheckDisk	622
LslCheckDiskH	623
LslFileDelete	624
LslFileDeleteH	624
LslFindCloseH	626
LslFindFirstH	626

LslFindFirst, LslFindNext, LslFindClose	628
LslFindNextH	628
LslFileTransfer	629
LslFileTransferFlush	629
LslFileTransferFlushH	629
LslFileTransferH	630
LslFileTransferFromPlcBuf	633
LslFileTransferFromPlcBufH	633
LslFileTransferMakeDir	635
LslFileTransferMakeDirH	635
LslFileTransferToPlcBuf	636
LslFileTransferToPlcBufH	636
LslFormatDrive	638
LslFormatDriveH	638
LslGetDiskSpace	639
LslGetDiskSpaceH	639
LslGetDriveListShort	641
LslGetDriveListShortH	641
LslRenameFileDir	644
LslRenameFileDirH	645
Parameters of a CB_FUNCTYPE Callback Function	646
Parameters of a CB_FUNCTYPE2 Callback Function	646
Managing the RefreshList	647
LslRefreshListAdd	657
LslRefreshListClear	658
LslRefreshListCreate	658
LslRefreshListCreateExt	659
LslRefreshListDestroy	660
LslRefreshListGetData	660
LslRefreshListGetDataSize	662
LslRefreshListGetLoaderVersion	664
LslRefreshListGetVarInfo	665

LslRefreshListIsOnline	665
LslRefreshListOffline	666
LslRefreshListOnline	666
LslRefreshListSetCallback	667
LslRefreshListSetData	669
LslRefreshListSymbolTableInit	671
LslRefreshListStartCount	671
LslRefreshListStart	672
LslSetDbgLevelEx	673
Internal Functions	673
LslDownloadOS	673
LslDownloadOSH	673
LslDirect	674
LslDirectCreateDOB	674
LslDirectDestroyDOB	674
LslDirectIsOnline	675
LslDirectOffline	675
LslDirectOnline	676
LslDirectReceive	676
LslDirectReceiveCount	677
LslDirectSend	677
LslDirectSetParam	678
Appendix A	678
LslExecNewInstr	687
LslExecNewInstrH	690
LslGetObjCls	691
LslGetObjClsH	691
LslGetObject	693
LslGetObjectEx	693
LslGetObjectExH	694
LslGetObjectH	695
LslGetObjectID	696

LslGetObjectIDH	696
LslReadFromClt	697
LslReadFromCltH	697
LslReadFromSvr	698
LslReadFromSvrH	698
LslReadFromSvrStr	700
LslReadFromSvrStrH	700
LslWriteToClt	701
LslWriteToCltH	701
LslWriteToSvr	703
LslWriteToSvrEx	703
LslWriteToSvrExH	703
LslWriteToSvrH	704
LslWriteToSvrStr	706
LslWriteToSvrStrH	706
Comlink API in the Loader	707
Introduction	707
The Comlink Interface in LASAL Class	707
Assigning CAN Identifiers	709
InstallCallBack	710
LDR_SetCanWait	711
LDR_SetCanWaitRemote	711
LDR_SetRs232ComlinkParams	712
Login	713
StartStopRefresh	715
TxCommand	715
TxCommandEx	716
TxUpd	717
Error Codes	718
Library for SIGMATEK Hardware Access	719
General	719
Installation	719
Requirements	719

CAN APIs	719
StkCanAddObject	720
StkCanClose	721
StkCanOpen	721
StkCanRxObjectAny	722
StkCanRxObject	722
StkCanSetup	723
StkCanSwReset	724
StkCanTxObject	725
DIAS APIs	726
StkDiasCheckError	726
StkDiasClose	726
StkDiasGetModID	727
StkDiasOpen	727
StkDiasReadByte	728
StkDiasReadCtrl	728
StkDiasReadWord	729
StkDiasReadRegister	729
StkDiasWriteByte	730
StkDiasWriteCtrl	730
StkDiasWriteRegister	731
StkDiasWriteWord	731
API Safety DLL	732
General Information	732
Interface Function ISAFETY_DLL	733
SAFETY_CHANGE_PASSWORD	734
SAFETY_CHECK_DONGLE_FW	735
SAFETY_CLEAR_CONFIG	735
SAFETY_DELETE_STATE	736
SAFETY_DOWNLOAD_FILE	736
SAFETY_DOWNLOAD_FW	737
SAFETY_FILE_GET_PRJNAME	738
SAFETY_FILE_GET_REVNBR	738

SAFETY_FILE_GET_SCPUNAME	739
SAFETY_GET_CFGSTATE	740
SAFETY_GET_IMAGE_MOD_ID_FW	740
SAFETY_GET_IMAGE_VERSION_FW	741
SAFETY_GET_MODUL_VERSION_FW	742
SAFETY_GET_RUNSTATE	742
SAFETY_GET_SAFETY_NBR	743
SAFETY_LEAVE_SERVICE_MODE	744
SAFETY_LOGIN	744
SAFETY_NEW_STATE	745
SAFETY_OPEN_CONNECTION	746
SAFETY_QUIT_ERROR	747
SAFETY_SET_CONFIGURED	748
SAFETY_SET_FILE	748
SAFETY_SET_IMAGE_FW	749
SAFETY_SET_SAFETY_NBR	750
SAFETY_SET_SERVICE_MODE	750
SAFETY_SET_TEMP_SERVICE_MODE	751
SAFETY_SET_USERPROMPT_TIME	751
SAFETY_SET_VERIFIED	752
Procedure Project Download	753
Procedure Firmware Download	754
Error Codes	755
FTP Server	757
Instructions	757
Quick Start	757
Requirements	757
Installation	757
Security	757
User Names	758
Passwords	758
Access Rights	758
Anonymous FTP	759

Server Configuration	759
ANONYMOUS	760
ANONYMACCESS	760
NOFTPSINI	761
WINCOMPLISTFORMAT	761
PASVPORT	763
DEFROOTDIR	763
MAXSESSIONS	764
FORWARDSLASH	764
BACKSLASH	764
NODRIVE	765
CMDTMO	765
WRITETMO	766
READTMO	766
TPRIO	767
DPRIORIO	767
Command Line Interface (CLI) Commands	768
FTPSVR INFO	768
FTPSVR HELP	768
FTPSVR START	768
FTPSVR OPTION	768
FTPSVR ACCESS	770
FTPSVR USER	771
FTPSVR STOP	773
Lasal OS FTP-Server API	773
OS Interface Library Class _FTPServer	773
AddAccess	774
AddUser	775
EditAccess	777
EditUser	778
GetConfig	780
GetUpdateFTPSiniStatus	782
GetUserAccessInfo	783
GetUserExtendedInfo	785

GetUserInfo	786
RemoveAccess	787
RemoveUser	788
StartServer	789
StopServer	790
SetConfig	791
UpdateFTPSini	792
Quick Review	793
FTPS.INI	793
Error, Warning and Info Logging	794
Appendix	795
Types	795
Error Values	797
Flags	798
Logfile	799
General	799
A Detailed Breakdown of Individual Log Entries	799
System	799
USB	807
DIAS	808
S-DIAS	812
VARAN	820
FTP	828
VNC	829
Log19	829
CPU Status and Error Messages	831
Troubleshooting DIAS Bus Problems	839
Troubleshooting for VARAN Bus Problems	842
Error Tree	842
Hardware Checklist	843
SIGMATEK Device Configuration	846
General Information	846
The LOGIN window	847

Areas in the Device Configuration	848
Title bar	853
Home	855
Settings	856
Network	857
Can Bus	861
Display	863
Config Files	866
Users	869
Logout	870
User Management	873
The "Service" user	874
The "Standard" user	877
Troubleshooting	878
Disclaimer	879

LASAL OS

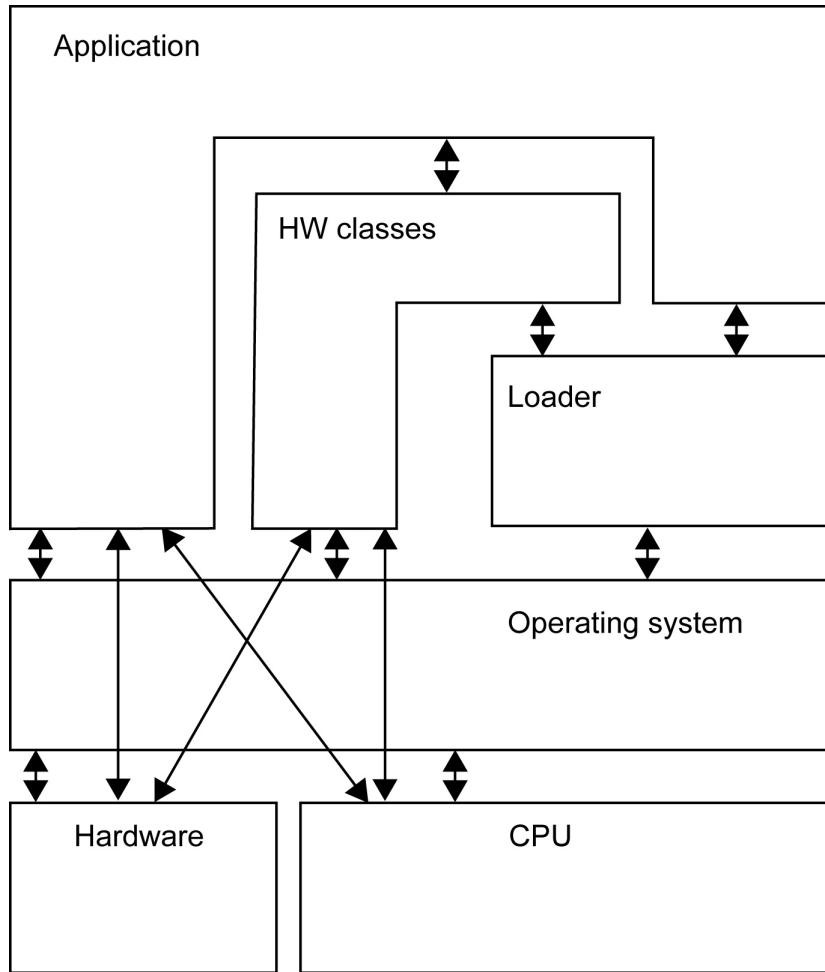
1 User Guide

1.1 User Guide

1.1.1 Software Structure of a Lasal CPU

A LASAL CPU is comprised of the following software modules:

- Operating system
- Loader
- Hardware classes
- Application



(* Graphic software section of a Lasal CPU *)

The interface between the individual modules is shown by the arrows.

1.1.2 The Loader

The loader is a component of every application and is found in each LASAL Class project. When LASAL Class is installed, the loader is also installed simultaneously and linked automatically to the LASAL Class project.

The loader's tasks are as follows:

1) Initialization

When the application is started, the initialization function is called from the loader. Once the initialization is complete, the cyclic part of the application (Background, Cyclic, Realtime) is called.

The initialization phase divided into the following sections:

- Initialization of the memory area for null voltage protected data.
- Assignment of classes and objects.
- Create client/server connections
- Allocation of the Init values.
- Enter the cyclic tasks in the operating system Update-List.
- Calling of constructors
- Calling of Init-methods

The init-methods of each object are called 12x. For the first 11 calls, the global variable `_firstscan` is 0 and is then set to 1 on the last call.

2) Cyclic Task

After initialization, the cyclic part of the loader is called by the cyclic task in 1 ms intervals for IPCs and 2 ms intervals for DIAS CPUs.

The cyclic task of the loader is comprised of the following parts:

Communication with other CPUs (Visu<->CPU, Comlink)

Instructions are first read, and then answered. Data is also sent without external request when, for example, the server value in the refresh list is changed.

Communication with the programming tool

A programming tool communicates with the debug interpreter to, for example, query server values.

Calling the instruction interpreter code (.IPR Files)

The runtime for the interpreter code is limited to a maximum value per scan (3 ms).

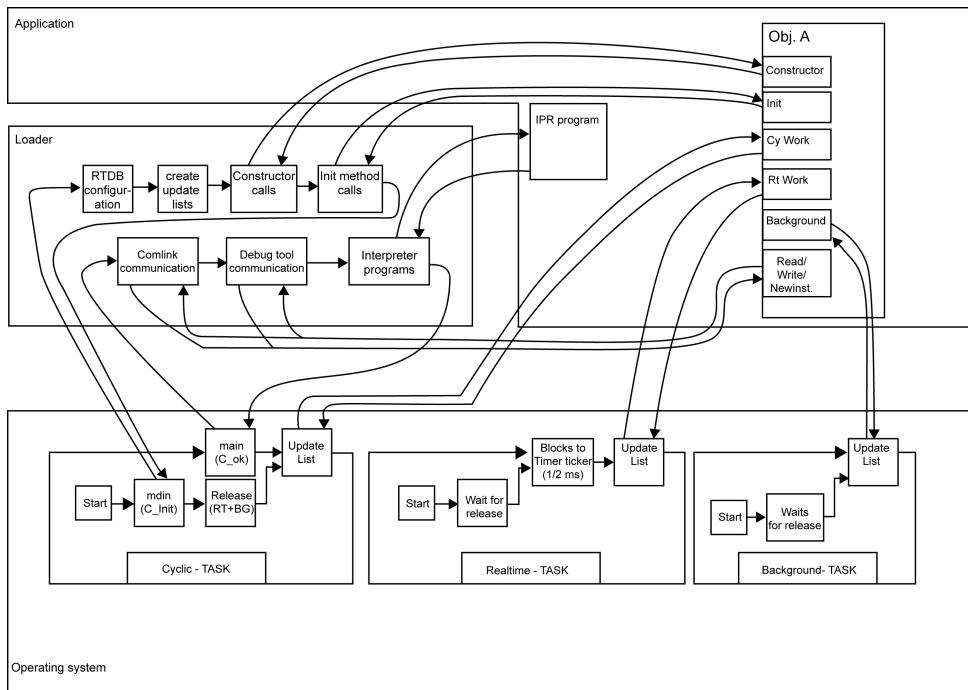
3) Functions Library

Several functions exist that are not available to the user but designed for Sigmatek classes.

- Administration of the null voltage protected data (RAM- und RAMEX- cells)
- APIs for the Comlink protocol (data exchange bet. Visu and PLC)
- Access the RTDB (data structures for classes and objects)
- Floating point functions

The ST compiler generates a function call with floating point operations. Floating point functions are available that contain the code for the math coprocessor and functions that the coprocessor copies. When floating-point functions are exchanged, the loader project must be changed accordingly and re-linked to the project.

The diagram below shows when and by which task the individual objects of a method are called.



(* Graphic call of object methods *)

Process from start of an application to the display of the server values

- 1) If the project is not yet in memory, the OS loads it from the hard drive or Memo module.
- 2) The project is linked, if not already.
- 3) The initialization phase of the loader is then called.
- 4) The cyclic tasks (CyWork-, RtWork and Background Tasks) are released which means the operating system processes the Update-list. When an object is found whose cyclic task has not been called within the preset time, the operating system calls the CyWork, RtWork or Background methods. An Update-list contains an entry for each CyWork, RtWork and Background task in which information such as the this-pointer, cyclic time, and time of the last call is stored. A list is managed for each type of cyclic task (Cyclic,

Realtime, Background). The cyclic part of the loader is called once before each scan of the cyclic Update-list.

The LSE kernel is initialized in the _LSE objects background method. The initialization of the kernel is comprised of:

- Loading the LSE project (static section only)
- Obtaining the Lasal ID (over the Comlink interface)
- Building the stat. Refresh list
- Synchronizing and loading alarms
- User interface

During the LSE initialization phase, progress bars are shown on the display screen. In this time frame, the CyWork and RtWork methods are already running.

6) Cyclic part of the LSE Kernel

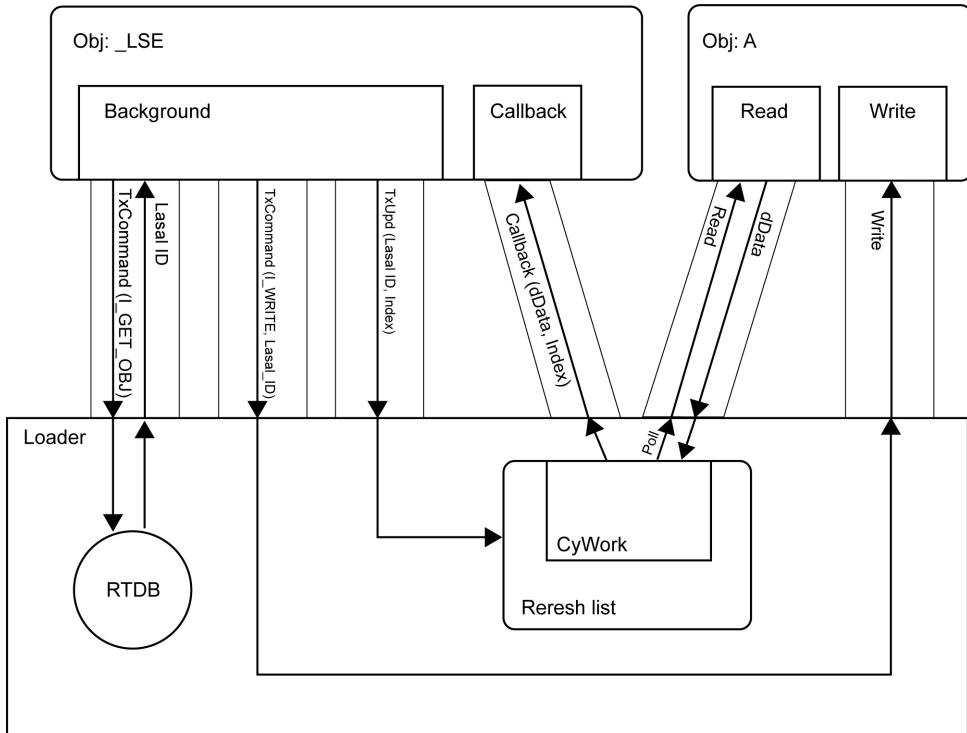
- Loading pictures + outputs.
- Construction of a dynamic list.

1.1.3 Comlink Interface

The visualization retrieves the server values over the comlink interface. The API functions used to call the comlink instructions are implemented in the loader.

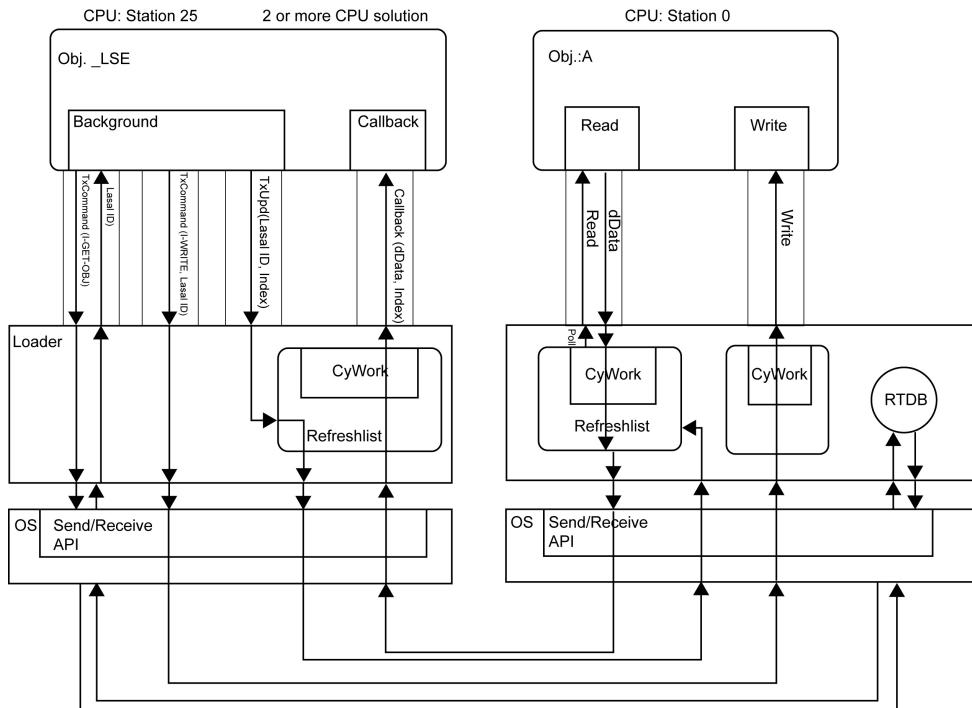
A comlink instruction sends a request to a target station, which can be its own CPU (1 CPU solution) or to a CPU connected over CAN or TC/IP (multiple CPU solution).

Comlink interface 1 CPU solution



(* Graphic comlink interface 1 CPU solution *)

Comlink Interface



(*Graphic Comlink Interface multiple CPU solution; two or more *)

The LSE kernel addresses the target station over a logic station number. The assignment of logic station numbers for physical addresses is stored in the C:\IPC.INI file.

To acquire a server value, the following comlink instructions are necessary:

- With **TxCommand**, the LasalID is queried using a server name.

The server value can be queried directly using the LasalID (polling), or a server in the refreshlist can be entered. When a server in the refreshlist is entered, the value of the server is periodically queried and the value is registered at the opposite station. There are 2 refresh lists: one static and one dynamic. The static is used by the LSE for servers whose values are continually needed (i.e. time). When changing screens, the dynamic refresh list is queried regularly and contains the server for the active screen.

With the **TxCommand** function, an instruction can be sent, which accesses a local or remote station with a read or write method. When the station is local, the method is called

within the same task without going through a queue. With the LSE kernel, the background task is used.

For remote stations, the instruction is transferred to the target station's loader. Since the cyclic part of the loader is running in the cyclic task, the method is called from the cyclic task.

1.1.4 Updating the Operating System

Do not turn the PLC off while the system update is in progress!



1.1.5 Backup of Persistent Data

Sample Configuration

The following entries are required in the AUTOEXEC.LSL file:

```
SET RAM FORMAT NEW
```

File system

This chapter contains information on the file system such as the drive letter assignments and an overview of the directory tree in a PLC containing a file system.

1.1.6 Drive Letter Assignments

LasalOS assigns the drive letters as follows:

IPC, C-IPC

A	First normal floppy drive
B	OS Version < 5.42: Second normal floppy drive OS Version ≥ 5.42: USB floppy drive
C, D, ...	The drive letters for hard disks are assigned in this order

- All partitions of the master device on the primary IDE controller.
- Only for an IPC: The internal smartmedia drive.
- All partitions of the slave device on the primary IDE controller.
- USB disks. The first 4 drive letters are reserved for 4 devices with 1 logical unit (LUN), the next 2 drive letters for a device with 2 LUNs units and the next 4 drive letters for a devices with 4 LUNs.

When the internal smartmedia drive (only on an IPC) or the slave device on the primary IDE controller is physically not available one drive letter is still reserved for these devices. On a C-IPC no drive letter is reserved for a smartmedia drive.

ET261, ET321

A-B	Not assigned
C	RAM disk
D	Smartmedia drive

Example

IPC with one partition on the hard disk of the IDE primary master and a USB disk drive with one logical unit:

C	Primary partition of the hard disk
D	Reserved for the internal smartmedia drive
E	Reserved for one partition of the IDE primary slave disk
F	USB disk drive

IPC with one partition on the hard disk of the IDE primary master, one partition on the hard disk of the IDE primary slave and a USB device with two logical units:

C	Primary partition of the IDE primary master hard disk
D	Reserved for the internal smartmedia drive
D	Primary partition of the IDE primary slave hard disk
F	Reserved for the 1 st USB disk drive with one LUN
G	Reserved for the 2 nd USB disk drive with one LUN
H	Reserved for the 3 rd USB disk drive with one LUN

I	Reserved for the 4 th USB disk drive with one LUN
J	USB device, 1 st logical unit
K	USB device, 2 nd logical unit

IPC with 2 partitions (one primary and one logical drive in an extended partition) on the hard disk of the IDE primary master and a USB device with four logical units:

C	Primary partition of the hard disk
D	Logical drive in the extended partition of the hard disk
E	Reserved for the internal smartmedia drive
F	Reserved for one partition of the IDE primary slave disk
G	Reserved for the 1 st USB disk drive with one LUN
H	Reserved for the 2 nd USB disk drive with one LUN
I	Reserved for the 3 rd USB disk drive with one LUN
J	Reserved for the 4 th USB disk drive with one LUN
K	Reserved for the USB device with 2 LUNs, 1 st logical unit
L	Reserved for the USB device with 2 LUNs, 2 nd logical unit
M	USB device, 1 st logical unit
N	USB device, 2 nd logical unit
O	USB device, 3 rd logical unit
P	USB device, 4 th logical unit

C-IPC with 2 compact flash cards, each card has one partition. A USB floppy is connected:

B	USB floppy drive
C	1 st compact flash card
D	2 nd compact flash card

Overview of the Directory Tree

This chapter describes system files and important parts of the LasalOS directory tree.

System Files

C:\AUTOEXEC.LSL	This is a batch file that is executed at startup. It typically contains commands to configure the system (e.g. online parameters, mouse and touch settings) and commands to load and start the application.
C:\LSLSYS	The LSLSYS directory contains files that are part of the operating system, e.g. DLLs, cli_help.txt and cli.sml.
C:\LSLCMD	The LSLCMD directory contains command files e.g. batch-files or scripts.
C:\SYSMSG	This is the default path where message files are stored.

Project Files

All files that are part of an application are stored in the application root directory. Unless otherwise specified, the root directory of the application is C:\.

<appl-root>\ACTIVE.DAT	This file exists only on platforms where the persistent data is not stored in a static RAM. The file contains a copy of the persistent data and is created when the application ends, is reset or if the UPS detects a power failure. During startup, the operating system reads this file and copies the data to the RAM section where constant data is stored. ACTIVE.SAV is a backup of a previous ACTIVE.DAT. Note: constant data is not copied to this file when the UPS is not enabled and the power is switched off!
<appl-root>\SRAM.DAT	This file serves the purpose as ACTIVE.DAT. The difference between the two files is the internal data format. Storing constant data is much faster using SRAM.DAT than ACTIVE.DAT. SRAM.DAT rather than ACTIVE.DAT is therefore recommended. To use SRAM.DAT the SET RAM FORAT NEW command must be inserted into the AUTOEXEC.LSL. SRAM.DAT is only supported with LasalOS versions 5.28 or higher and the project is compiled with LASAL Class version 0.60 and above.
<appl-root>\SRAM.SAV	
<appl-root>\LSLWORK	The LSLWORK directory contains the object files (LOB-files) of the Application and a description of the project and its files (IDX-file).
<appl-root>\IPC.INI	The IPC.INI file is a configuration file for the LSE-Kernel.
<appl-root>\MPC	The MPC directory contains the files, used by the LSE-Kernel; such as MPC-files, fonts and bitmaps.

1.1.7 Logging of System and Application Messages

The Sysmsg facility enables the application to produce error, warning and other messages. It is often important that these messages can be written to a file and later viewed. Operating system messages and messages from an application are written to different files, which can be viewed to identify problems.

When a message is generated it is not immediately written to a file. Instead, it is first written to a buffer. The operating system message buffer is written to a file when a power-down is detected or a serious error occurs. The application buffer is written to a file when the application is stopped or a power down is detected.

It is important to activate the UPS when using the sysmsg facility. When the UPS is not enabled, the operating system has no chance to write the message buffer to a file if a power down occurs. If the UPS is not enabled or if no UPS exists, the message buffer is written to a file the next time power is applied. A limit, called a file quota, can be defined for the message file to avoid filling the disk completely. If the message size exceeds the file quota, the message is reassigned as a backup file and a new message file is created. When this occurs, however, an existing backup file is lost. The disk size therefore required for message files is 2 * file quota; the file name is EVENTxx.LOG, in which xx is a unique number for each message buffer object (the operating system message buffer number is 00). The path where the message file is stored can be configured using the environment variables SYSMSGPATH (operating system messages) and APPMSGPATH (application messages). The default path is C:\SYSMSG.

The sysmsg facility can be configured using the Command Line Interface (CLI) commands and the parameters specified in the Sysmsg API interface functions. The path to where the message is stored and the file quota can be configured. When a system has multiple disk files, more practical to set the path of the message files to a drive that does not contain the operating system.

The CLI commands needed to configure the sysmsg are SET EVENTLOG ON|OFF <file-quota>, SETENV SYSMSGPATH <path> and SETENV APPMSGPATH <path>. Please refer to the Command Line Interface for a detailed description of the CLI commands.

1.1.8 Operating System Messages

Operating system messages are generated when special events occur; each event has a time stamp at the beginning of the line that shows the date

04/10/01;

formatted as YY/MM/DD (year/month/day) followed by a semicolon and the time

13:32:00.692;

formatted as hh/mm/ss/ms (hours/minutes/seconds/milli-seconds), followed by a semicolon.

The time stamp then reads

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

Events, which must log more information, contain the current time stamp (may be changed in the future) in the first line only.

The log events caused by interrupts have a different format:

*I*01:DIAS Error (3F)

The line starts with

I

and identifies an interrupt followed by a number, which is incremented by one each time such a logging event occurs. This number is followed by a colon and the error text.

Some events need a stack dump in order to be logged. These lines are normally 16 hexadecimal values, each representing one byte. If the stack-pointer points to a memory region where access is denied, no useful values are available and the dump shows "???" for each value.

Not all logged events are of significance for the user! Some are only useful to the LasalOS developers.



Detailed list of events and their descriptions:

- explanation of time stamp is always omitted
- examples only where necessary (text varies)

System Messages

PowerON, OS [I] ([II])	[I] OS version [II] OS build date Each time the PLC is powered Example: 04/10/01;13:32:00.692;PowerON,OS:5.80 (Aug 12 2004, 12:55:00)
Reboot	Each time the PLC rebooted
USV Power down recognized	Each time the PLC recognizes a power-down condition
USV Power down finished	power-down action is finished

Battery low!	Battery-low interrupt occurred
Power fail!	Power-fail interrupt occurred
Over temperature	Over-temperature interrupt occurred
Temperature sensor error!	Temperature-sensor-error interrupt occurred
Task [I] is terminating!	<p>[I] taskname</p> <p>Occurs always when a system-task is terminated due to an exception.</p> <p>Example:</p> <p>Task TCP_SERVER0 is terminating!</p>

Error messages

System

SYSKERNEL_SP: ST_MailBox full	Service provider mailbox is full
SYSKERNEL_SP_Q ueue: not initialized	Service provider mailbox not (yet) initialized
SYSKERNEL_SP_Q ueue: Queue full	Service provider command-queue is full
TaskStopHook: SPQ not empty	Service provider command-queue not empty
Error (down)loading OS	Error on preparing LasalOS download
WATCHDOG Error	CPU watchdog expired
F A T A L S Y S T E M E R R O R	At this point the LasalOS is stopped. There are, eventually, some more lines with more detailed information.

Download OS

▪ No MEM Access	Error on preparing LasalOS download (CPU 386)
▪ No EPROM	Address of EPROM not found
▪ Image Error	Length of OS image wrong
▪ MAPMem EPROM	Access rights of EPROM cannot be adjusted

Error	
▪ OS Write Error	Error on writing OS Image to EPROM
▪ No RTB-File	No RTB file found
▪ invalid RTB-File	RTB file invalid
▪ invalid RTB-Filelength	Length of RTB file invalid
▪ RTB-FileOpen	Error on opening RTB file
▪ RTB-FileCopy	Error on writing RTB file to disk
▪ BOOTDisk Operation	BOOTDisk operation failed, disk not bootable

Debugger

Debug_Handler: Exit (unknown bignum); ip_addr= [I]	[I] EIP of breakpoint Occurs when the debugger encounters an invalid breakpoint Example: Debug_Handler:Exit(unknownbignum); ip_addr= 0x001a0fe4
----------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------

Runtime Error

RUNTIME Error;Task= [I] ;Object= [II]	[I] taskname [II] objectname Written each time a runtime-error occurs in the realtime, cyclic or background task. "taskname" and "objectname" define the task and object which caused the error
------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Application Heap Error

AppMemError	Various information
-------------	---------------------

TCP/IP

IP-Address conflict; [I] already used by another station in	[I] IP-address [II] response-time
-------------------------------------------------------------------	--------------------------------------

the network; response time [II]	Occurs on in initializing an Ethernet interface with an IP address which exists already in the network
error xn_set_ip: [I]	[I] error code Setting an IP-address was not successful

Exception

Exception in AppHeap(EIP: [I]), [II]	[I] EIP of error [II] Exception type An exception in a function managing the application-heap occurred; application will be terminated
Exception in ACTIVE functions(EIP: [I]), [II]	[I] EIP of error [II] Exception type An exception occurred on writing ACTIVE.DAT/SRAM.DAT; application will be terminated
Exception [I], EIP=[II]	[I] Exception type [II] EIP of error An exception occurred; application will be terminated

DIAS Error

DIAS Error ([I])	[I] DIAS station (hexadecimal) A DIAS Error occurred
DIAS Risc Error ([I])	[I] DIAS station (hexadecimal) A DIAS RISC Error occurred (illegal instruction, illegal user command)

Both "Exception" and "DIAS Error" are followed by the following information blocks:

Project name	[I]:CpStat=[II] [I] project name [II] project state
Hint whether exception or DIAS Error	[I] - Dump: [I] Exception or DIAS
Register dump	EIP:... EAX:... EBX:... ECX:... EDX:... ESI:... EDI:... ESP:... EBP:... EFL:... CS:... DS:... SS:... ES:... FS:... GS:...

Task, in which error occurred	<p>TASK=[I],STACK=[II]-[III],MIN=[IV]</p> <p>[I] Task name [II] Stack start [III] Stack end [IV] First not used stack address</p>
Memory data	<p>APPHEAP:Start=[I],Size=[II],Used=[III],Free=[IV]</p> <p>[I] Application heap start address [II] App. Heap memory size [III] App. Heap used memory [IV] App.heap free mem.</p> <p>USRCODE:Start=[I],Size=[II]</p> <p>[I] User code start address [II] User code size</p> <p>USRDATA:Start=[I],Size=[II]</p> <p>[I] User data start address [II] User data size</p>
Stack dump	<p>Example:</p> <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 list	<p>List all tasks</p> <p>Example:</p> <p>Main Task :00D00000-00DFFF88,1033360d</p> <p>Task name</p> <p>Tasks stack start address</p> <p>Tasks stack end address</p> <p>Used stack bytes</p>
Call stack	<p>Lists the call stack. Functions of LasalOS do not have names, they are listed with their EIP as "EIP= ...". Application or Loader functions are listed with their name and module.</p> <p>Example:</p> <p>CALL STACK:</p> <p>03792028:CLASSPRJ@.\cppcode\cppObjects\T_main.obj::_RTK_READY (00541) 03792048:CLASSPRJ@.\Lse_\Lse_00_00.stl::BACKGROUND@_\LSE (02010) 03792058:EIP=0x001AD57A 03792090:EIP=0x0017D952</p>

037920C4:EIP=0x0014A55D

Debug Messages

This messages can be configured by CLI commands

```
SET DBGLEVEL topic level
```

Topic

CANLL	Low level CAN routines
APPHEAP	Application heap
GRAPHIC	Graphics library
SYSHANDLER	Exception handle
CAN	CAN communication
CDIAS	CDIAS library
TASKLISTS	Task list
BREAKPOINTS	Breakpoint handling
FLASH	CIPC flash module
IP	TCP/IP communications
TGRAPH	LARS graphics
SERIAL	Serial interface library
USB	USB interface library (default level =2)
DEBUGIP	Loader: Debug Interpreter
MODLOAD	Module loader
KERNEL	LasalOS Kernel (Maintask, Serviceprovider)
LINKER	Linker library
LOADER	Loader library
TASKS	Task handling
VNCCLI	VNC client
VNCSVR	VNC server
LSLFILE	LslFile library

LSLCMD LslCmd library

Level

- | | |
|---|--------------------------------------|
| 0 | No logging |
| 1 | Error logging |
| 2 | Additional debug information logging |
| 3 | Extended debug information logging |

Sample Configuration

Given a Compact Flash card containing the operating system (C:\), a hard disk in which messaged should be stored (D:\) and a maximum operating system file of 2 Mb. The operating system messages should be stored in D:\SYSMSG and the application messages in D:\APPMMSG.

The following entries are the required in the AUTOEXEC.LSL file:

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

1.1.9 Runtime Check and Watchdog

LASAL has implemented the following runtime checks:

- CDIAS peripheral reset
- DIAS slave peripheral reset
- Intelligent master watchdog
- Cyclic runtime check
- Background runtime check (since LasalOS 01.01.008)
- Realtime runtime check (since LasalOS 01.01.008)
- Error detection

This applies to all CPUs (CIPC, CCL911,...).

1.1.10 Hardware Watchdogs

There are three hardware watchdogs that supervise the peripheral devices, If not triggered on time, the peripheral devices will reset. There are 2 phases in which the watchdogs are triggered:

Init phase	In this phase, the hardware and hardware classes are initialized. The watchdogs need to be triggered for initialization; this is done by the OS.
RunRam phase	Here, the hardware is completely initialized and the application has taken control over the devices. Here, the hardware classes trigger the watchdog.

The watchdogs are not triggered during the other phases (power up, booting the OS, reset status,...). The hardware watchdogs time-out every 130 ms, while the software trigger is set in increments of 100 ms. This value was chosen to provide extra time and allow some software jitter, however, it can be set to accommodate user requirements.

CDIAS peripheral reset	The CDIAS device contains a watchdog circuit, which initiates a peripheral reset if the watchdog circuit is not triggered within 130 ms. The CDIAS watchdog is triggered by the OS in the Init phase and the hardware classes during RunRam status.
DIAS peripheral slave reset	A DIAS slave device contains a watchdog counter that generates a reset when the bus is not accessed within 130 ms. The watchdog is triggered by the OS in the "Init"-phase and the hardware classes during RunRam status.
Intelligent master watchdog	The intelligent master contains a watchdog counter that switches off the intelligent master if no access to a special register is detected within 130 ms. The watchdog is triggered by the OS in the Init phase and the hardware classes during RunRam status.

1.1.11 Software Checks

In addition to the hardware watchdogs, there are several runtime-checks that supervise the Real time, Cyclic and Background tasks. If the runtime for a task (the total runtime of the respective methods) exceeds a preset value, the PLC stops the application and resets the peripheral device (because no watchdog is triggered).

Cyclic Runtime Check

The operating system continuously scans the list of objects with a CyWork method and calls the CyWork method when the method's call-time has expired. When scanning this list exceeds the pre-defined time, the program is stopped and a RUNTIME (CPU Status Code 2) error is triggered.

The cyclic runtime check can be set as follows:

1. CLI command 'SET RUNTIME <time in 10 ms units>'

2. For CPUs without a CLI (Command Line Interface), the global variable `_swruntime` contains the value for the cyclic runtime check in 10 ms units

A value of 0 disables the cyclic runtime check.

Background Runtime Check

The operating system continuously scans the list of objects with a CyWork method and calls the Background method when the method's call-time has expired. When scanning this list exceeds the pre-defined time, the program is stopped and a BT-RUNTIME error is triggered (CPU Status Code 68).

The background runtime check can be set as follows:

1. CLI command 'SET BTRUNTIME <time in 10 ms units>'
2. For CPUs without a CLI (Command Line Interface), the global variable `_swbruntime` contains the value for the cyclic runtime check in 10 ms units

A value of 0 disables the background-runtime-check.

Real time runtime check

When the duration of all RtWork methods is longer than the real time interval, the program is stopped and a RT-RUNTIME error (CPU Status Code 67) is triggered. The duration of the Realtime interval can be set with the CLI command SET RTRUNTIME.

1.1.12 Error Detection

If the LasalOS detects an error, the application will be terminated (there is currently one exception, the "DIAS Error").

"DIAS Error"

The CLI command, SET DIASERROR OFF, disables this error condition. If the "DIAS error" is enabled with CLI command, SET DIASERROR ON, the default setting of the PLC terminates the application and resets the peripheral devices.

The "OS_SSR_InstallDIASHandler" Macro allows the application to install a "DIAS error" handler, which is called when "DIAS error" is enabled and the error occurs. If the handler returns "0" (zero), no error condition will be generated and the application continues to run, in all other cases, the application is stopped and the peripheral devices are reset.

A DIAS error can be caused three ways:

- DIAS general failure (access to device not present)
- DIAS RISC error

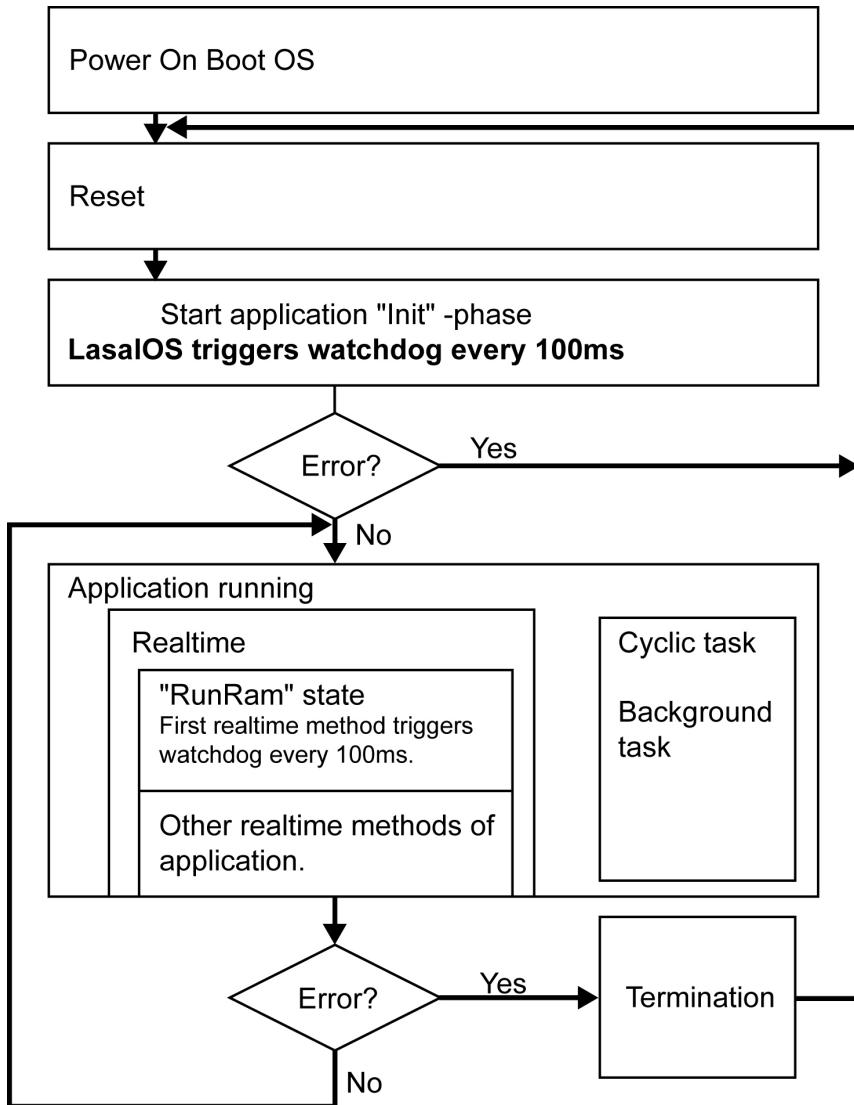
- DIAS unknown opcode error

Other Error Conditions

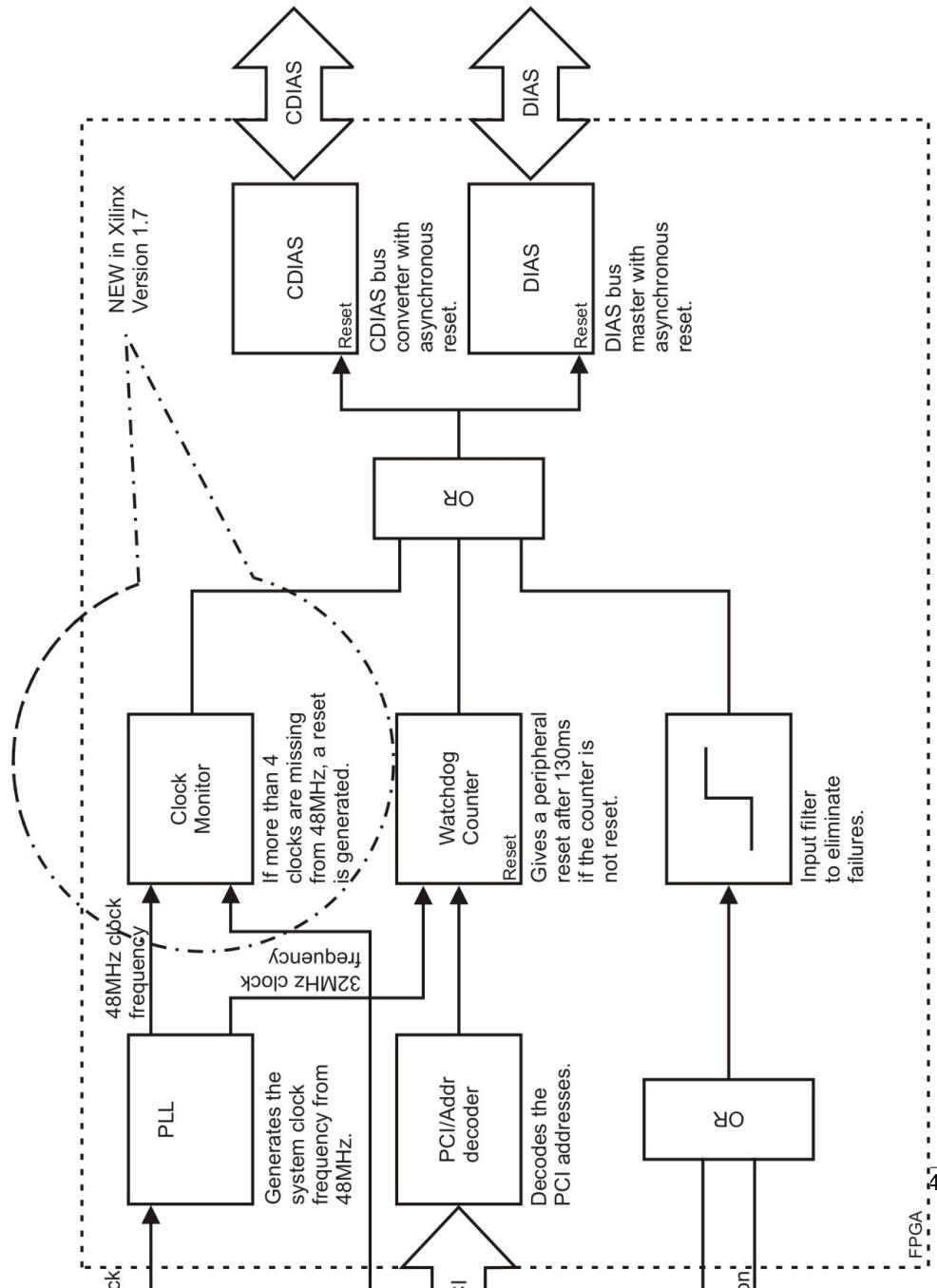
The application is terminated and the peripheral devices reset with the following errors:

1. CPU Exceptions
 - Divide error, overflow, invalid opcode, coprocessor not available, double fault, coprocessor segment overrun, invalid TSS, segment not present, stack exception, access exception, page fault;
 2. LOADER error
 - If the LOADER encounters errors (out of memory, unknown class id, unknown constructor, unknown object, unknown channel, wrong connection, wrong attribute, syntax error, virtual file error, class incompatible, class must be updated) the state is unchanged (no change to state reset because ACTIVE.DAT would be written with wrong data) but the watchdogs no longer triggered. Therefore the peripheral devices will reset. Real time, Cyclic and Background tasks are not running in this state.
 3. HEAP error
 - "APPLMEM ERROR" terminates the application, watchdogs are not triggered, peripheral devices will reset.
 4. Errors due to optional debug facilities
 - HEAP
- With CLI command "SET DBGHEAP ALL ON" one can generate additional conditions to create an "APPLMEM ERROR" with more details in the event-log.
 - Application is terminated, watchdog is not triggered, peripheral devices will reset.

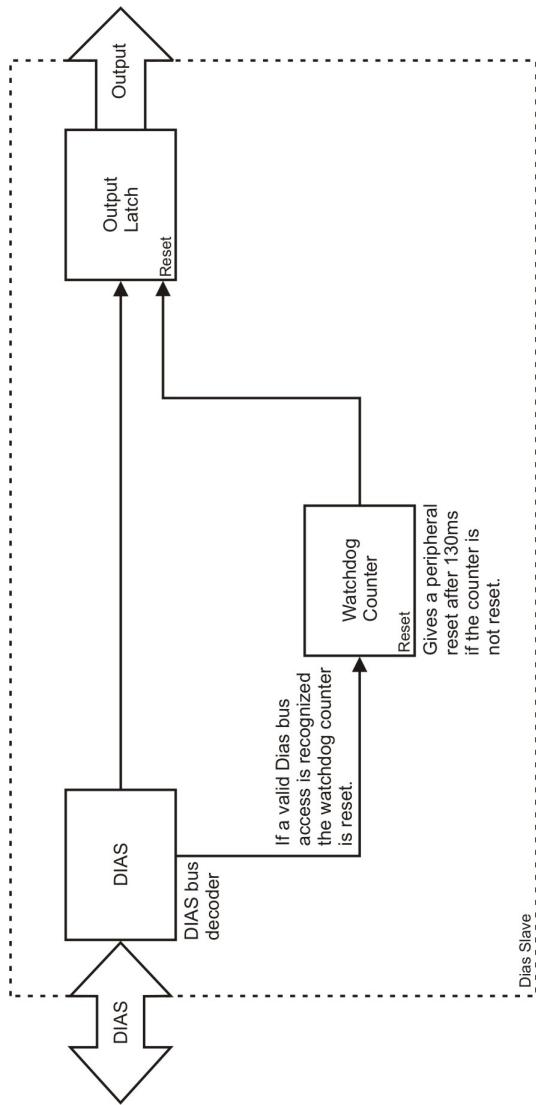
1.1.13 Flow of Operation



1.1.14 CIPC Peripheral Reset



1.1.15 DIAS Slave Peripheral Reset



1.1.16 Simple Router

To use an IPC or a CPU as a simple router, 2 network interfaces must be available. The two network interfaces cannot be in the same logic network, which means that the IP address of both interface connections must be configured so that the first cannot reach with the second.

Such a configuration, however, is not possible.

Using the autoexec.lsl to set the IP-Addresses, a conflict with the second address forces the Lasal OS not to install the second interfaces.

The command line interface shows a message on the screen (if available).

Example

LASAL Class Online Connection from a Windows PC to an IPC via a CIPC Simple Router.

Windows PC

```
IP Adresse: 10.100.100.100
Subnet: 255.255.255.0
Gateway: 10.100.100.150
```

The Windows PC must have a gateway. If a packet is sent over the network and no other host (another PC in the same network) can reach the packet, it will be sent to the gateway.

The gateway (router) is an accessible host in the network with a second network interface to forward the packet.

The above explanation serves only to provide an understanding of how the router behaves, the windows PC knows that no host can reach the packet and sends directly to the gateway.

CIPC

2 network interfaces (router/gateway)

network interface 1: autoexec.lsl settings:

```
IP-Address: 10.100.100.150  SET IP HOSTADDR 10 100 100 150
Subnet: 255.255.255.0  SET IP SUBNET 255 255 255 0
Gateway: no  SET IP GATEWAY 255 255 255 255
```

network interface 2: autoexec.lsl settings:

```
IP-Address: 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: no  SET IP 2 GATEWAY 255 255 255 255
```

IPC

autoexec.lsl settings:

```
IP-Address: 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
```

With this configuration it's possible to connect from a PC with IP-Address 10.100.100.100 to an IPC in a different network with IP-Address 192.168.43.154.

The windows PC will be send a packet to 192.168.43.154. This address cannot be in the same logical network as the win. PC.

So the packet will be send to the gateway 10.100.100.150. This address is the first interface of the simple router. The network interface 1 internally forwards the packet to the second network interface with the address 192.168.43.155.

The second network interface sends the packet to 192.168.43.154.

1.1.17 Security Downgrade

This function can be used to prevent loading an invalid (old) operating system version from a USB stick.

Activate Security Downgrade

- add the entry, "min_update_version=x" to the "C:\lslsys\config.lsl" file
- whereby x is the version number (min_update_version=2)
- create a file named "version.lsl" on the USB stick
- add the entry "has_version=2" (without ") to this file
- only when has_version is greater or equal to min_update_version is the script on the USB stick executed.

1.1.18 LASALOS Supported USB Devices

These drivers are available for all CPU's except CCL722, DCL642 and DTC081.

Hub driver All standard-hubs are supported.

Keyboard driver The keyboard driver supports all USB keyboards using USB class 3 (HID), subclass 1 (boot device) and protocol 1 (keyboard). Up to four keyboards can be operated simultaneously. Additional keyboards are ignored.

Mouse driver	The mouse driver supports all USB mice using USB class 3 (HID), subclass 1 (boot device) and protocol 2 (mouse). Up to four mice can be operated simultaneously. Additional mice are ignored. Includes also HAMPSHIRE-Octopus touch controller.
Printer driver	The printer driver supports all USB printers using USB class 7 (Printer), subclass 1 or 2 (uni- or bidirectional), and any protocol. Up to four printers can be operated simultaneously. Additional printers are ignored.
Disk driver	The USB disk driver supports all USB disks which adhere to the USB Mass Storage Device Specification, class code 8, subclass 4 (UFI), 5 (FDD), or 6 (SCSI), protocols 0 (Control/Bulk/Interrupt), 1 (Control/Bulk/Interrupt, no Interrupt status) or 0x50 (Bulk only). Any number of devices each with up to 2 terabyte capacity is supported. Supported are also devices with integrated hubs and/or multiple LUNs.
USB to serial	Up to 4 devices "ATEN UC232A" are supported. They are available as COM7-COM10.
Barcode reader	The Barcode Readers "CipherLab 1067", "CipherLab 1090" and "CipherLab 1021" with HID interface are supported. They are treated as keyboards, the data read is available as a stream of keys. No special software beside an input-field is required.

1.1.19 LASALOS Supported USB Devices for CCL722, DCL642 and DTC081

Disk Driver

The USB disk driver supports all USB disks which adhere to the USB Mass Storage Device Specification, class code 8, subclass 4 (UFI), 5 (FDD), or 6 (SCSI), protocols 0 (Control/Bulk/Interrupt), 1 (Control/Bulk/Interrupt, no Interrupt status) or 0x50 (Bulk only). Only one drive is supported. The drive may not contain an embedded hub. If the drive contains multiple LUN's, only the first one is supported.

USB-to-Ethernet Driver

The USB-to-Ethernet driver supports all USB-to-Ethernet adapters with the following chipset:

- ASIX AS88772
- ASIX AS88172

The following products have been checked:

- SIIG, Inc - USB 2.0 to 10/100 Ethernet, Model Number US2277

- D-Link - USB 2.0 Fast Ethernet Adapter, Model Number DUB-E100

This driver is not available on all platforms.



1.1.20 USB Update

Requirements

The USB stick has to be connected to the CPU directly (not with a HUB) and should not have a integrated HUB. For controls with an RTK operating system, only one USB stick, namely the update stick, may be connected to the control during the update process.

1. Project Update

To perform a project update it is necessary to create a “bootdisk” on a USB stick.

To do so, perform the following steps:

- Connect the USB stick with your PC
- Load the project in LASAL CLASS
- Rebuild the project
- Click “Create Bootdisk...” in the project menu
On the appearing dialog select the USB drive and click OK.

Additional to this you have to create a autoexec.lsl file in the root directory of the stick containing the following commands:

```
LSLOAD F:  
LSLSAVE
```

Now you can connect the USB stick with the CPU and reboot.

During the boot operations the project will be loaded to the internal Flash.

2. Update LSE Project

To update the LSE project you have to copy all files from the runtime directory of the LSE project to the stick and perform some file operations in the autoexec.lsl:

```
MKDIR C:\MPC           REM Create directory if not exists  
XCOPY F:\LSE\*.* C:\MPC\  REM copy project files
```

There are also other file operations available: DEL, RMDIR, REN and FORMAT.



The C: drive on the CPU is the buffered flash disk drive, while F: is the USB drive.

3. OS Update

To perform a OS update the following command is necessary in autoexec.lsl:

```
BOOT F:\<file name>
```

<file name> is the name of the BIN file (Example. CCL722.BIN) on the USB stick

Connect the stick to the CPU and reboot. During the boot operations the OS will be updated. The update is completed when the CPU is in state OS INSTALLED. Disconnect the USB stick and reboot once more to boot the new OS.

Sample autoexec.lsl for 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

To perform an update of the autoexec.lsl, an existing autoexec.lsl file must be copied from an USB stick to the root directory of the CPU (C:).

The following command is necessary:

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

If the autoexec.lsl, which is executed from the USB stick, exists within the same directory as the autoexec.lsl that should be copied to the root directory of the CPU, a different name has to be used.

```
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 HTML Files

Web pages can be transferred from the CPU to a browser and represented from the browser through an available Web Server on the CPU.

This is not available on all platforms!



The documentation [LASALOS Web-Server](#) contains a detailed description of the configuration and usage of the Web-Server.

To perform an update of the HTML files, all files have to be copied to the CPU. All directories and subdirectories must exist (must be created if they do not exist).

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

The XCOPY command supports wildcards (*, *.* , *.txt, ...) but does not work recursive (no subdirectories).

6. Support for Slow USB Sticks

Depending on the manufacturer, the initialization times of USB sticks can vary strongly.

For this reason, it is possible that while the PLC is starting, the USB stick may be detected late or not at all; even though it has been inserted into the USB port.

As a result the autoexec.lsl file on the USB stick may not be found during startup and therefore not processed.

To support such USB sticks, the startup time can be increased by a configurable time.

To configure the startup time, „bootStickDetectionDelayTime=x“ must be inserted into the C:\lslsys\config.lsl file. Whereby, x is the time in milliseconds by which the startup time is extended. The maximum allowable value for the entry is 15000 ms.

The startup time is generally increased via the above-mentioned entry, regardless of whether the USB stick is actually inserted or not.

1.1.21 How to Setup/Defragment a Compact Flash Medium

Install a LasalOS

To install a LasalOS version on a Compact Flash you need the Windows™ tool "bootdisk.exe" and the RTB or LBI of the desired operating system. Both are available on request at Sigmatek support. At the command-prompt you execute the following command.

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

os	the file name of the desired operating system
ext	RTB or LBI
x:	the drive letter of your compact flash

Format a Compact Flash

Under some circumstances, if all files should be deleted for example, it may be necessary to format the Compact Flash. The Compact Flash must be formatted correctly with at least 4 sectors per cluster.

The "format" instruction in LasalOS does this automatically. To format a Compact Flash with Windows™ you must enter the following command at the command prompt.

```
format /A:2048 /FS:FAT x:
```

x:	the drive letter of your compact flash
----	----------------------------------------

Defragmenting a Compact Flash

A Compact Flash medium can be defragmented with a Windows™ operating system. After defragmentation, it is necessary to reinstall the LasalOS as described above.

1.1.22 Lasal OS Tasks

Task names	Short info
Main Task	task for processing the ServiceProvider queue: commands to start and stop the application through the online connection and through the OS_SSR_AddToKernelSP from the application
Idle Task	idle task
CPU Monitor	idle task (calcs the CPU time)

DLED_Task	to update the leds
RT_AsyncTask	task for the asynchronous file methods from the _FileSys class
PANELTASK	task for keypad, leds und command line interface (DTC281)
CMD_LINE	command line interface
KEY_TASK	task for the keypad (CET281, BDF2000)
LCD_TASK	refresh-task for LCD
CANx_Stationx	CAN online connection
CanLL Timer	timer task for CAN protocol
CanRxMailThread	task for all incoming CAN messages
CANx_Listener	task for all incoming CAN messages (LARS)
PLC386_TASK	to set the baudrate, ... during startup through the SET key, setting the leds, loading application
Execute Task	to execute all functions through the EXEC commandos called with the function OS_SSR_AddToKernelSP
HiPrio Task	to stop the application because of errors in an interrupt routine (DIAS error)
UserLogTask	task for the asynchronous writes of the eventlog files
KEYLED_TASK	task for the keypad and leds (e.g. ET261)
RT_Runtime	highpriority task to stop the application because of an Realtime Runtime Error
TCP Client x	TCP online connection
TCP Server	TCP server for the online connection
IPTASK	Internal task of the IP stack (per network interface)
INTERRUPT	Internal task of the IP stack (per network interface)
TIMER	Internal task of the IP stack
DbgRxFast	serial online connection
TASK_DBGSERWIN 32	serial online connection (LARS)
SSFDCTask	task for the smartmedia driver
USB Hub	USB hub
USB Mouse	USB mouse
USB Keyboard	USB keyboard

USB Touch	USB touch
USB Print	USB printer
KEYSHUTTLE_TASK	task for the keypad and the shuttle wheel
VNC SERVER	VNC server
WSP0_CT	cyclic task
WSP0_RT	realtime task
WSP0_BT	background task

1.1.23 Assignment of CAN Identifiers

1.1.23.1 PLC (online + comlink protocol)

The following CAN identifier ranges are reserved for the comlink protocol:

- 16#700 – 16#71F
- 16#500 – 16#67F (provided that the default value for the number of max. comlink channels is not changed)

The identifiers of the CAN objects that are actually allocated depend on the following factors:

- CAN station number of the PLC (STATION), range: 0-31
- Number of outgoing comlink connections (NBR_CONNECTIONS)
- Max. number of comlink channels (MAX_CHANNELS). The default value is 5. This value can be customized by changing the #define COMLINK_CAN_COMCHS in the loader (Lasal1: loader.h, Lasal2: UserDef.h). After changing this value, the loader has to be compiled and linked with the application. Note that all PLCs that communicate over the comlink protocol have to use the same value !

These are the formulas to calculate the allocated 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

```

Example

3 PLCs with station numbers 0, 11, 25.

Station 11 and 25 establish a comlink connection to station 0.

The default value for COMLINK_CAN_COMCHS is untouched (-> 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
-> Allocated CAN Objects: 16#700, 16#70B, 16#719, 16#584, 16#585, 16#62C, 16#62D
```

1.1.23.2 MiniDisplay

An instance of a MiniDisplay class also uses defined CAN identifiers dependent on the user-configured station identification (via the client NodeNumber).

receiving object	16#180 + n
sending objects	16#200 + n, 16#300 + n, 16#400 + n, 16#500 + n

n specifies the station identification.

1.1.24 Enumerating Ethernet Interface Connections

Since a PLC can support more Ethernet interfaces it was agreed to number them following a fixed scheme.

1	First internal interface (ETH1)
2	Ethernet interface at the EWP (or ETH2 at the CCL912)
3	Varan interface
4-6	reserved
7	USB to Ethernet adapter
8	Second USB to Ethernet adapter

1.1.25 Status and Error Messages

During the status test for the LASAL Class software, status and error reports are given. If the device is a CPU with a status display, the status and error messages are also shown here. POINTER and CHKSUM errors are also displayed on the terminal screen.

In addition to displaying "Er" and a code from 0 to 255, any one of the following combinations can occur:

Pr Ln	currently linked
Er Ln	linking error
Er Ld	project is loading
Er OP	Opsys is programming
Er AL	cause of error / status unknown
Er Lo	Loader error. There is no further information on the cause of the error. The application may have a problem (unknown object, unknown connection, etc.)

Number	Report	Meaning	Reason/Remedy
00	RUN RAM	The user program is currently running in the RAM The display is not affected.	
01	RUN ROM	The user program stored in the memory module has been loaded into the RAM and is currently running. The display is not affected.	
02	RUNTIME	Total duration of all cyclic objects exceeds maximum time; the time can be configured	

		using 2 system variables: <ul style="list-style-type: none"> • Runtime: time remaining • SWRuntime: Pre-selected value for runtime counter 	
03	POINTER	Incorrect pointers were located before running the user program.	<p>Possible causes</p> <ul style="list-style-type: none"> • Memory module is missing, not programmed or damaged. • Program in user program memory (RAM) is not executable. • The battery buffer has failed. • A software error in which the user program is overwritten. <p>Solution</p> <ul style="list-style-type: none"> • Reprogram the memory module. If the error persists, exchange the module. • Exchange buffer battery • Correct program error
04	CHKSUM	An incorrect check sum was detected before running the user program.	Cause/solution: See POINTER
05	WATCHDOG	The program was interrupted by the watchdog logic.	<p>Possible causes</p> <ul style="list-style-type: none"> • User program interrupts blocked over a long period of time (STI command forgotten). • A Hardware interrupt was programmed incorrectly. • INB, OUTB, INW, OUTW commands used incorrectly • Processor damaged <p>Solution</p> <ul style="list-style-type: none"> • Correct program error • Exchange CPU
06	GENERAL ERROR	GENERAL ERROR	
07	PROM DEFECT	An error occurred while programming the memory module.	<p>Possible causes</p> <ul style="list-style-type: none"> • Memory module damaged

			<ul style="list-style-type: none"> • User program too big • Memory module missing <p>Solution</p> <ul style="list-style-type: none"> • Exchange memory module
08	RESET	CPU has received a RESET command and is waiting for further commands. The user program is not processed.	
09	WD DEFECT	The watchdog logic is defective. After startup, the CPU checks the watchdog functions. If an error appears during the test, the CPU deliberately enters an infinite loop and accepts no further commands.	<p>Solution</p> <ul style="list-style-type: none"> • Exchange CPU
10	STOP		
11	PROG BUSYS		
12	PROGRAM LENGTH		
13	PROG END	The memory module has been programmed successfully.	
14	PROG MEMO	The CPU is currently programming the memory module.	
15	STOP BRKPT	The program was stopped by a breakpoint.	
16	CPU STOP	The CPU was stopped by the PG-software (F6 HALT at the status test).	
17	INT ERROR	The CPU has triggered the wrong interrupt and stopped the user program, or has encountered an unknown command while running the program.	<p>Possible causes</p> <ul style="list-style-type: none"> • An operating system command was used, which does not exist. • Stack error (Odd number of PUSH and POP commands). • User program interrupted by a software error. <p>Solution</p> <ul style="list-style-type: none"> • Correct program error
18	SINGLE STEP	CPU is in SINGLE STEP mode and waiting for further commands.	

19	READY	A module or project has been sent to the CPU and is now ready to execute the program.	
20	LOAD	The program is stopped while the CPU is receiving a module or project.	
21	UNZUL. MODUL	The CPU has received a module that does not belong to the program.	
22	MEMORY FULL	The operating system memory (heap) is too small. No memory could be reserved by calling an internal or interface function from the application.	
23	NOT LINKED	A missing module or a module that does not belong to the project was detected during the CPU startup.	
24	DIV BY 0	A division error has occurred.	<p>Possible causes</p> <ul style="list-style-type: none">• Division by 0• The quotient does not fit in the result register. <p>Solution</p> <ul style="list-style-type: none">• Correct program error
25	DIAS ERROR	An error has occurred while accessing a DIAS module.	<p>Possible causes</p> <ul style="list-style-type: none">• An attempt was made to access a nonexistent DIAS module• DIAS bus error <p>Solution</p> <ul style="list-style-type: none">• Check DIAS bus• Check terminating resistors
26	WAIT	CPU is busy.	
27	OP PROG	The operating system is currently being reprogrammed.	
28	OP INSTALLED	The operating system has been reinstalled.	
29	OS TOO LONG	The operating system cannot be loaded due to insufficient memory.	
30	NO OPERATING SYSTEM	Boot loader message.	

		No operating system found in RAM.	
31	SEARCH FOR OS	The boot loader currently searching for the operating system in RAM.	
32	NO DEVICE		
33	UNUSED CODE		
34	MEM ERROR	The operating system installed is not designed for the hardware.	
35	MAX IO		
36	MODULE LOAD ERROR	The LASAL module or project couldn't be loaded.	
37	GENERAL OS ERROR	General error while loading operating system.	
38	APPLMEM ERROR	Error in the dynamic application memory administration (user heap).	
39	OFFLINE		
40	APPL LOAD		
41	APPL SAVE		
46	APPL-LOAD-ERROR	Error while loading application.	
47	APPL-SAVE-ERROR	Error while saving application.	
50	ACCESS-EXCEPTION-ERROR	Read write access of restricted memory area, e.g. writing on NULL pointer.	
51	BOUND EXCEEDED	Exception error at memory area infringement.	
52	PRIVILEGED INSTRUCTION	Prohibited command for current CPU level, e.g. setting of segment register.	
53	FLOATING POINT ERROR	Error during a floating-point operation.	
60	DIAS-RISC-ERROR	Error from intelligent DIAS master.	
64	INTERNAL ERROR	Internal error, all Applications stopped	Restart, Message to Sigmatek
65	FILE ERROR	Error during file operation	
66	DEBUG ASSERTION FAILED	Internal error	Restart, Message to Sigmatek
67	REALTIME RUNTIME	Total time of all Realtime objects exceeds maximum time; this value cannot be can not	From version 1.1.7

		be configured: 2 ms at 386er CPUs 1 ms all other CPUs	
68	BACKGROUND RUNTIME	Total time of all background objects exceeds maximum time, time can be configured with 2 system variables: <ul style="list-style-type: none">• BTRuntime: remaining time• SWBTRuntime: Preselection value for runtime counter	
70	C-DIAS ERROR	A connection error with a C-DIAS module has occurred.	<p>Possible causes</p> <ul style="list-style-type: none">• The cause of the error is documented in the log file <p>Solution</p> <ul style="list-style-type: none">• This depends on the cause
72	S-DIAS ERROR	A connection error with a S-DIAS module has occurred.	<p>Possible causes</p> <ul style="list-style-type: none">• real network does not match the project• S-DIAS client is defective <p>Solution</p> <ul style="list-style-type: none">• analyze logfile
75	SRAM ERROR	An error occurred while initializing, reading or writing SRAM data.	<p>Possible causes</p> <ul style="list-style-type: none">• SRAM configured incorrectly• SD card formatted incorrectly• SD card removed• battery for the supply of the internal program memory is empty <p>Solution</p> <ul style="list-style-type: none">• evaluate log file (Event00.log)• check configuration• change battery for the supply of the internal program memory• format SD card as EDGE medium with LASAL Class 2

			• check SD card
95	USER DEFINED 0	User definable code	
96	USER DEFINED 1	User definable code	
97	USER DEFINED 2	User definable code	
98	USER DEFINED 3	User definable code	
99	USER DEFINED 4	User definable code	
100	C_INIT	Start of initialization, configuration is processed.	
101	C_RUNRAM	LASAL project has been started successfully from RAM.	
102	C_RUNROM	LASAL project has been started successfully from ROM.	
103	C_RUNTIME		
104	C_READY	Everything is okay	
105	C_OK	Everything is okay	
106	C_UNKNOWN_CID	Unknown class from a stand-alone or embedded object; or unknown base class.	
107	C_UNKNOWN_CONSTR	Operating system class cannot be created, probably wrong operating system.	
108	C_UNKNOWN_OBJECT	An unknown object was found in an interpreter program. Creation of more than one DCC080 object.	
109	C_UNKNOWN_CHNLL	Number of HW module bigger than 60.	
110	C_WRONG_CONNECTION	No connection to required channels.	
111	C_WRONG_ATTR	Wrong server attributes.	
112	C_SYNTAX_ERROR	No specific error, recompile all project sections and reload all.	
113	C_NO_FILE_OPEN	Tried to open an unknown table.	
114	C_OUTOF_NEAR	Memory assignment failed.	
115	C_OUTOF_FAR	Memory assignment failed.	

116	C_INCOMPATIBLE	Object with the same name, but another class already exists.	
117	C_COMPATIBLE	Object with the same name and class already exists, but has to be updated.	
224	LINKING	Application is currently linking	
225	LINKING ERROR	Error while linking, message in LASAL status window.	
226	LINKING DONE	Linking finished	
230	OP BURN	Operating system is stored in flash memory	
231	OP BURN FAIL	Error while burning of operating system	
232	OP INSTALL	Operating system is currently being installed	
240	USV-WAIT	Supply has been turned off, UPS is active.	
241	REBOOT	Restart of operating system.	
242	LSL SAVE		
243	LSL LOAD		
252	CONTINUE		
253	PRERUN	Application is started.	
254	PRERESET	Application is finished.	
255	CONNECTION BREAK		

1.1.26 Routing

1.1.26.1 Overview

- CPUs can be networked together so that each CPU can act as a gateway to any other CPU.
- The addressing is done via identifiers (0..254).
- The network can be set up as a hybrid, meaning that it is irrelevant whether CAN or Ethernet or both are used.
- The routing table is provided to the application via a system variable.
- Routing works for the Lasal32 protocol only.
- Aliases are possible.

- Routes must be unique.
- Routing is not supported via RS232.
- There can only be a single route over one CAN line.
- Supported CPUs are: C-IPC, CCL91x, CCL722, DCL642, DTC081, DTC081_IP, ET261, ETT321, IPC and BDF2000

1.1.26.2 Commands

set station <n>

Sets the station identifier for a CPU. The identifier must be set before setting the routing table! "set station," without an additional parameter shows the currently set identifier.

set routingtable <filename.ext>

Loads the routingtable from the specified filename. Only the entries which are relevant to the CPU are interpreted (specified by the CPU's identifier). The routing table will be cleared when the command is issued without a parameter.

1.1.26.3 Routing Table Syntax

The file is analyzed line by line; entries are not case sensitive. Incorrect entries as well as those not intended for the current CPU are ignored. It may be prudent to check the output of this command.

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

n	0-254
TargetIP	IPv4 address
Port	default Port 1954
TargetCAN	0-31 (CAN station number)
Local	The target station is identical to the local station (Alias)

„;“, „#“ and „rem“ identify comments.

Example

```
; Station0
STATION01      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     local
```

Each station decides what to do with a packet independently. When a station gets a packet to route with an unknown target station, the packet is discarded and an error message returned.

1.1.27 Boot Screen

Platforms that support a color graphic interface and have a file system can show a user-defined start screen instead of the CLI.

The user-defined screen must be stored in c:\bootpic.bmo so the CPU can locate it.

The file can be generated using the “bmp2bmo.exe” tool included in the LASAL Screen Editor.

The boot image is unloaded by either of the following event:

- Application starts (if there is no _LSE the standard screen will be shown)
- <TAB> or <ESC> is pressed during booting
- Online command is received from Lasal during booting
- Reset command from Lasal

An activity bar overlays the screen to inform the user, that the CPU is busy. Furthermore a message will appear above the activity bar after the autoexec.lsl has ended.

Trace messages will be suppressed. If there is an exception within the application the CPU will display the CLI.

Situations, which need user attention:

Problem	The message “Autoexec.lsl ended!” does not appear but the activity bar is alive.
Reason	There may be a “PAUSE” command or any other command, which expects a user input (which is obviously invisible during the display of the start picture).

Solution	Remove either the boot picture or the interactive commands from the autoexec.lsl.
Problem	There is a static screen shown.
Reason	CPU may have crashed or the touch screen is not properly calibrated.
Solution	Press <ESC>, if there is no response check your autoexec.lsl and remove the boot picture temporarily to see what is going on during booting. If the CLI is shown properly, identify the interactive command e.g. "set mouse type hamusb" with a missing touch configuration file and start it manually.

Be aware that a boot picture shall only be used on systems which have cleared development state! A temporarily removal of the boot picture can be accomplished by either renaming the file "bootpic.bmo" to something else or by setting the CLI command "bootpic exit" at a desired place within the autoexec.lsl.

1.1.28 File System for USB Sticks

Analysis suggest, that it may be advantageous to format a USB stick with a FAT32 file system instead of a FAT16 file system to gain speed by a considerably large amount.

1.2 Task Management

1.2.1 Threads

In the LASAL OS, threads are run in parallel. Whereby, a code section is processed independently and have their own register image and separated stack. Threads are also referred to as tasks, but they should not be confused with LASAL tasks (LASAL tasks are described in the following chapter).

Since in a single processor system, different threads cannot be executed at the same time, the processor switches forth and back between the tasks. In order to ensure that a currently running thread reaches the CPU when no other thread with a higher priority exists, the scheduler decides which thread should be activated. If the scheduler has to decide between several threads with the same priority, it selects the thread that has the longest wait time.

The scheduler is called when an event occurs that changes the thread status. For example, a thread changes to ready status (executable) after a message is received. The scheduler is called in regular intervals, which are called Timer Ticks. The value of Timer Ticks is not the same on all platforms. The Timer Tick indicates in which time interval the LASAL tasks are started. For an IPC the value is 1 ms, for a 365 processor this value is 2 ms. The Timer Tick can be set on the current platform ('Set Maintimer'). Here, values between 125 µs and 2000 µs are possible.

There are two types of threads: application and operating system threads. Each LASAL task category (real-time, cyclic and background) has its own application thread (real-time, cyclic and background).

The operating system threads are responsible for initialization, monitoring, time management and assigning services, as well as online communication.

The following table provides an overview of the most important threads (tasks) in LasalOS.

Thread (Task)	Task	Priority
Realtime	Processing of the RtWork LASAL task	16
OS kernel task	Runtime monitor (cyclic, background) processing of operating system tasks	15
Cyclic	Processing of the CyWork LASAL tasks	14
Communication task	Online communication (operating system)	14
Background	Processing of the LASAL background task The priority of the background tasks depends on the VISU LOW HIGH setting (LOW: 10, HIGH: 14)	10/14

1.2.2 LASAL Tasks

LASAL tasks are methods of a LASAL object (RtWork, CyWork or Background), which are cyclically called by the operating system. If the application is initialized, the loader writes the LASAL task in the task list. Task lists are for real-time, cyclic and background tasks. Each task list entry contains a time interval (period), which can be configured in LASAL classes (or objects).

The application tasks are processed in an infinite loop in order to check every entry (object), as to whether the time interval has elapsed since the last call. If the specified time has elapsed, the method is called and the time point of the last call is updated.

After the task list is complete (application ended / in Reset / in Error), the application tasks are allowed to continue for a specified time in order to process the task list. The real-time task is run for a time period of 1 ms in IPCs and 2 ms for 386 CPUs. With current

platforms, this value is configured (Tick: 125 µs - 2 ms). The time period for the cyclic task is a minimum of 1 ms, but can be higher when the tick is higher. The background task is normally 2 ms.

VISU LOW/HIGH

The visualization (LSE) is executed in the background task. The visualization is negatively affected, when the CPU has a high cyclic or real-time load, as the background has a lower priority by default. With the VISU HIGH setting, the background task can be raised to the same priority as the cyclic task.

The CLI command SET VISU LOW | HIGH can be used to change the priority setting from low to high.

LASAL Task Runtime:

Since LASAL tasks in the same category are processed sequentially, the runtime of a LASAL task affects the call-time of the other LASAL task within the category. The runtimes of the LASAL tasks should be as short as possible, in order to maintain the configured cycle times.

If an operating system call is located in a LASAL task, which blocks the application thread, The LASAL tasks in the corresponding category are also blocked. The LASAL task is only processed further when the application thread can run again. Operating system calls, like TCP/IP functions or multi-thread interface functions. In an RtWork method, this call should be avoided.

Control and Monitoring

The processing of LASAL tasks can be ensured (monitored) via runtime monitoring. This function is configured in the autoexec.lsl or the CLI with the following commands:

set rtruntime Value	For real-time monitoring – value can contain 0-255 and is specified in maintimer ticks. The duration of a maintimer tick is set with the CLI command SET MAINTIMER.
set runtime Value	For cyclic monitoring – value can contain 0-255 and is specified in 10 millisecond increments (e.g. 30 corresponds to 300 ms).
set btruntime Value	For background monitoring – value can contain 0-255 and is specified in 10 millisecond increments (e.g. 30 corresponds to 300 ms).

The value 0 deactivates monitoring of the according category. If processing a task requires longer than the time specified in the runtime monitor, the application is stopped and a runtime error triggered.

The execution times for the LASAL tasks can be monitored with the following variables:

- _RealMaximumTime

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

The `MaximumTime` variable indicates the maximum runtime for a task from the start of the project (or from the last reset of the variable). The `AverageTime` variable contains the average value of a task list's runtime, whereby the value is calculated from the last cycle with 50%, the second to last with 25% etc. The variables are specified in microseconds and are defined in the `Rtos_variables.h` file.

Locking Interrupts

Locking interrupts can have a negative impact on the time properties of the LASAL tasks. The timer interrupt and subsequently, the scheduler call can be delayed. The delay of the timer interrupt call changes the time frame of the real-time task; this increases jitter.

If the interrupts are locked for more than 1 timer tick, the timer interrupts are lost and the operating system delays the internal clock. The real-time calls are then lost and the applications contain the wrong time values.

1.3 Programming Guide

1.3.1 The Construct

The constructor is called by the loader when creating an object and can be used to initialize the object data elements.

There are two constructors available:

- Standard constructor
- User-defined constructor

Standard Construct

The standard construct code is generated automatically and is therefore of no concern to the user. One of its functions, for example, is to create the method list for virtual methods. Calling the user-defined construct, if available, is the last task. The standard construct name is `STD`.

User-defined Construct

The user-defined construct can be created by the end-programmer. The method name for the user-defined construct is the same as the class name. If a construct is addressed in the following document the user-defined construct is meant

Construct Function Prototype

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

The construct return value causes a loader error when the value does not equal C_OK; the project is therefore not started.

The following should be noted when using the construct

An object cannot be accessed over a client channel if other objects or connections to other objects exist.

If a client is connected to an OSI (Operating System Interface) class, this channel can be used since this connection is established before constructor is called.

The start values for the private data elements of the object are initialized before construct call. If a private data element is initialized in the construct, the object's start value is overwritten.

Operating system calls not made over an OSI class can be called using a macro (OS_SSR_xxx).

1.3.2 Exception Handling

An “exception” is an event that is unexpected or disrupts the ability of the application to proceed normally. Exceptions can be detected by both hardware and software.

The application can install an exception handler, which can respond when exceptions occur. What happens after the handler is run depends on its return value. The handler can do one of the following things:

- Pass control to the system handler (usually stops the application immediately).
Return value: 0 (EXCEPTION_OSHANDLER)
- Handle the problem and resume normal operation.
Return value: -1 (EXCEPTION_CONTINUE_EXECUTION)

- Skip execution of the current object and continue with the next object.
Return value: 1 (EXCEPTION_SKIP_CONTINUE)

The last two options (-1 and 1) do not stop the application and option 1 is not available under all circumstances. It can only be used when an exception occurs in a cyclic method. Code is run in a cyclic method when it's called directly or indirectly from one of the cyclic methods (CyWork, RtWork or BackGround). Examples of code not used in a cyclic method are the construct of an object, the Init-method, Read- or Write methods called from the loader or a task created with OS_MT_CREATETHREAD.

1.3.3 APIs

1.3.3.1 OS_SSR_SetHandler

OS_SSR_SetHandler allows an application to install a handler routine, which is called when an exception occurs.

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;
```

Transfer parameters	Type	Description
handlerType	DINT	Specifies the type of the event that should trigger a handler call; currently only HANDLERTYPE_EXCEPTION is supported
pFct	pVoid	Pointer to a handler function with a __cdecl calling convention. The prototype of this function is as follows: <pre>FUNCTION __CDECL EvalException VAR_INPUT excptCode : UDINT; excptInfo : ^EXCEPTION_POINTERS; taskType : UDINT; pThis : pVoid; param : pVoid; END_VAR</pre>

		<pre>VAR_OUTPUT retVal : DINT; END_VAR</pre> <p>excptCod identifies the type of exception that occurred. e Possible values that can appear are EXCEPTION_ACCESS_VIOLATION, EXCEPTION_INT_DIVIDE_BY_ZERO etc. (see Isl_st_ifssr.h)</p> <p>excptInfo is a pointer that contains information about the exception, e.g. register values at the time of the exception. This pointer can be NIL when no information is available!</p> <p>taskType is the task during which the exception occurred. Possible values are TASK_OBJ_CT, TASK_OBJ_RT etc. (see Isl_stitask.h).</p> <p>pThis is a pointer to the object that executed the code during the exception. This parameter can be NIL when no object information is available (e.g. when a User Task generated an exception).</p> <p>param is the parameter passed to OS_SSR_SetHandler.</p> <p>retVal defines what happens after the handler is triggered. Possible values are EXCEPTION_OSHANDLER, EXCEPTION_CONTINUE_EXECUTION or EXCEPTION_SKIP_CONTINUE.</p>
param	pVoid	This value is passed to the handler function. The OS does not interpret this value, it is simply passed on and can be used to pass arbitrary information to the handler.
Return parameters	Type	Description
ret0	DINT	

1.3.3.2 OS_SSR_SkipOverIDIV

OS_SSR_SkipOverIDIV can be called from a handler routine in the event of a division error to skip the division and resume with the next instruction.

Macro

`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;

```

Transfer parameters	Type	Description
exceptionCode	UDINT	Identifies the exception type. The value of this parameter should be the same as the exception code passed to the handler routine.
exceptionInfo	pVoid	This pointer contains information about the exception. The value of this parameter should be the same as the exception information pointer passed to the handler routine.
quotient	UDINT	The value of this parameter is the quotient when the division is skipped.
remainder	UDINT	The value of this parameter is the remainder when the division is skipped.
Return parameters	Type	Description
retVal	DINT	The return value is EXCEPTION_CONTINUE_EXECUTION when the division can be skipped or EXCEPTION_OSHANDLER in the event of an error. Set this value to the return value of the handler routine.

Example

```

#include "LSSL_ST_EXCPT.H"
#include "LSSL_ST_IFSSR.H"

VAR_GLOBAL
    udQuot  : UDINT;
    udRest  : UDINT;
END_VAR

FUNCTION VIRTUAL GLOBAL NewClass0::Init

    IF _firstscan THEN

        IF _LSSL_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 Overview

1.4.1.1 File System Commands

<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 Operating System Commands

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

1.4.1.3 LASAL Data File (active.dat) Commands

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

1.4.1.4 Kernel Error Log Commands

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

1.4.1.5 Projects Commands

SET FPUROUNDCONTROL	LSLOAD / LO	LSLSAVE / SA	PROJECT
RESET	RUN		

1.4.1.6 Configuration Commands

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

1.4.1.7 Configuration Commands (IP)

IP	SET IP		
--------------------	------------------------	--	--

1.4.1.8 Configuration Commands (CAN)

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

1.4.1.9 Configuration Commands (DIAS)

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

1.4.1.10 Configuration Commands (SERIAL)

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

1.4.1.11 Configuration Commands (Mouse/Touch)

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

1.4.1.12 Configuration Commands (Printer)

<u>SET PRINTER</u>			
------------------------------------	--	--	--

1.4.1.13 Batch Processing Commands

<u>CLS</u>	<u>DELAY</u>	<u>ECHO / @ECHO</u>	<u>EXIT</u>
<u>GOTO</u>	<u>IF EXISTS</u>	<u>PAUSE</u>	<u>REM</u>

1.4.1.14 Environment Variables

<u>FILENAMES</u>	<u>REBOOTONPF</u>		
----------------------------------	-----------------------------------	--	--

1.4.1.15 Command Line Syntax

A CLI (Command Line Interface) command, including its required or optional parameters and switches, must be written in the following standard format:

COMMAND required parameter [optional parameter] [/switch]

All items included in square brackets are considered optional.

1.4.1.16 Keys and Files

F2	Repeats last key at this position
F3	Repeats last command from this position
ESC	Clears current input line
BSPC	Deletes last input character
CTRL-ALT-F1	Sets US keyboard mapping
CTRL-ALT-F2	Sets German keyboard mapping
cmd > filename	Redirects command output to file
file.bat	Executes batch file
AUTOEXEC.LS L	Startups batch file (searches A:\, then C:\)
?	Displays all CLI commands

1.4.1.17 Order of Execution of Commands in autoexec.lsl

When the LasalOS is running an autoexec.lsl file during boot-up, not all commands are executed in the order in which they appear in the file. Several commands are queued internally and then executed after the autoexec.lsl has been processed. The queued and non-queued commands are executed in the order as they appear in the autoexec.lsl.

The queued commands consist of:

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

The following example requires attention.

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

Here, an attempt is made to install a new boot image and then pause the execution. The [BOOT](#) command is queued, so [PAUSE](#) is executed before [BOOT](#). This process is different from what the two lines suggest.

To avoid this error, an extra command named [QUCMD](#) is called that queues a command:

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

With this command, [BOOT](#) is queued automatically and [PAUSE](#) is queued after [BOOT](#) so that it is executed after the [BOOT](#) instruction.

1.4.2 File System Commands

The commands [CD/CHDIR](#), [COPY/C](#), [CREATE](#), [DEL/ERASE](#), [DIR/D](#), [FC](#), [MD/MKDIR](#), [RD/RMDIR](#), [REN](#), [TYPE](#) and [XCOPY](#) require at least one parameter for the path or file name. Starting from LasalOS V01.01.049, the path or filename can be put in quotes ("") and embed spaces, e.g.

```
DEL "c:\path with blanks\filename with blanks.txt"
```

Path and file names created with blanks through LasalOS are always treated as long file names. The short names automatically created by the systems do not contain blanks and are somewhat different from the file names created by other operating systems.

The environmental variable, [FILENAMES](#), can be set to LONG so that the [DIR](#) command shows the long file names.

1.4.2.1 ARCH

The command is used to archive and unzip files and folders.

Short form: ARCH

Syntax

```
ARCH <filename>
-r<root-dir> - source directory of the file to be archived
-f<file-spec> - file name of the file to be archived (wildcards are supported)
-x - unzips an archived file
-s<split-size> - splits the file into several small files on
```

Example 1

Archiving

In the following example, a single document is archived and stored in a different destination folder. The "text.doc" file to be archived, is located in the folder "C: \ DocFolder

\" and the archive should be created in the folder "C:\ZipFolder\" under the name "TextZip.zip".

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

To archive an entire folder structure, the same command without file specification is used.

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

Unpacking

To unzip the archive "TextZip.zip" in the folder "C:\ZipFolder" to the folder "C:\UnzipFolder", enter the following command:

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

Example 2

Archiving

In the following example the entire folder structure is archived and saved in another target folder. The files to be archived are in the folder "C:\DocFolder" and the archive should be created in the folder "C:\ZipFolder" under the name "TextZip". The archive should be split in single parts with a size of 1 000 000 bytes with the option -s. Hint: the single parts of the archive will be slightly bigger than 1 000 000 bytes, as each parts gets a header.

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

Unpacking

To unpack the split archive "TextZip" in the folder "C:\ZipFolder" to the folder "C:\UnzipFolder", enter the following command:

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

Requirements

The command requires the DLM lslzip.dlm.

1.4.2.2 ATTRIB / ATTR

The Attrib command is used to display, set or remove one or more of the four attributes (read-only, archive, system, and hidden) that can be assigned to files and directories. It is typically used to remove read-only, hidden, and system attributes so a file can be moved, deleted or set so that neither is possible.

Short form: ATTR

Syntax

To display the attributes of a directory.

```
ATTRIB directoryname
```

To display the attributes of a file.

```
ATTRIB filename
```

To set or remove attributes of a file or directory

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

- | | |
|---|--------------------------|
| + | Sets an attribute |
| - | Clears an attribute |
| A | Archive file attribute |
| H | Hidden file attribute |
| R | Read-only file attribute |
| S | System file attribute |

Example

To display the attributes of a file named NEWS86 located on the current drive, type the following command.

```
ATTRIB news86
```

To assign the Read-Only attribute to the file REPORT.TXT, type the following command.

```
ATTRIB +r report.txt
```

1.4.2.3 CD / CHDIR

Changes the current default directory.

Short form: CD

Syntax

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

Example

Either of the following commands changes your current directory to the directory named LSLWORK.

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

Suppose you have a directory named MPC with a subdirectory named IMAGES. To change your current directory to \MPC\IMAGES, type the following command.

```
cd \mpc\images
```

Or, if your current directory is \MPC, you can use the following command to change to the \MPC\IMAGES directory.

```
cd images
```

To change from a subdirectory back to the parent directory, type the following command.

```
cd..
```

To change to the root directory, type the following command.

```
cd\
```

1.4.2.4 CHKD SK

Checks the file-system of the specified drive against errors and optionally corrects them.

Syntax

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

drive: - the drive-letter of the drive to check

/F - if specified, tries to correct the errors

Example

The following command checks drive C:

```
Chkdsk C:
```

The following command checks drive D: and corrects errors.

```
Chkdsk D: /F
```



This command is used to check a drive's file system for errors. Some errors can be corrected with option /F. At times, it may be necessary to run chkdsk with option /F twice to clear errors. If the file system still contains errors, it is no longer reliable and should be reformatted and initialized.

1.4.2.5 **COPY / C**

Copies a file to a specified destination without checking for existing files.

Short form: C

Syntax

```
COPY source [destination]
```

source - File name of the file to be copied.

destination - File name and/or directory of the new file, default is ".".

Example

```
copy memo.doc letter.doc
```

The following command copies a file:

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

To copy a file named ROBIN.TYP from the current drive and directory to an existing directory named BIRDS that is located on drive C, type the following command:

If the BIRDS directory does not exist, LASALOS copies the file ROBIN.TYP into a file named BIRDS that is located in the root directory on the disk in drive C.

Remarks

The LasalOS tries to allocate the destination file in adjoining disk sectors to reduce disk fragmentation and improve file access speed. If not possible, the text message "Information: file not contiguous" is displayed. Although this is neither an error nor warning message, it is an indication that the disk is either heavily fragmented or rather full.

Another message that can appear is "SRAM full". This is a real error message, which means that the destination file could not be created and indicates that the destination disk needs to be reformatted.

Both topics are described in the LasalOS User Guide, chapter "How to setup/defragment a Compact flash medium".

1.4.2.6 **CREATE**

Creates a new file with defined size but undefined contents.

Syntax

```
CREATE filename file-size allocated-size
```

filename - File name of the file to be created

file-size - Actual length of the file

allocated-size - Allocated length of the file, contiguous

Example

The following command relates a file.

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

The file has an actual length of 0 bytes and occupies a reallocated space of about 20 Mbytes.



- The content of the file is always undefined.
- This command is used for performance reasons:
- The seek operations are reduced on HD and FDD.
- On the C-IPC with files in committed mode the FAT has not to be rewritten after changing the actual file-size; only the directory has to be updated.
- If the file-size exceeds the pre-allocated size it will be extended in a standard, not contiguous, way and performance decreases because of seek-operations and FAT updates.

1.4.2.7 DEL / ERASE

Erases a file.

Short form: DEL

Syntax

```
ERASE filename
```

filename - Name of file to erase.

1.4.2.8 DIR / D

This command outputs a list of files and directories.

Short form: D

Operating System Rtk

DIR without parameters shows as header line the drive and if necessary the name of a directory. E.g. The command "dir mpc" creates the header "Directory of C:\mpc\".

Below this, one line per directory or file is displayed with the information described below.

For a directory:

Name plus extension, identification "<DIR>", date and time when the directory was created, the attributes.

For a file:

Name (8 characters) plus extension (3 characters), file size in bytes, date and time of the last change of the file contents, the attributes.

The footer displays the total number of files and their total size in bytes.

Syntax

DIR [drive:] [/W] [/X] [/P] [/<attr>+|-] [path | file name]

/W - Displays only the names in wide format, with up to five file names and directory names in each line. Directory names are enclosed in square brackets.

/X - Extended display. The header additionally displays the volume label. For files, the required space on the storage medium in bytes is also displayed, as well as the first cluster number of the file and the number of cluster chains. In the footer lines, the number of free bytes on the storage medium is also displayed.

/P - Pause after each full screen page.

/<attr>+|- - include/exclude attributes. Valid values for <attr> are: H, S, R, A, V, D.

path | file name - The paths and file names to be displayed can contain the wildcards (*,?).

Operating System Salamander

DIR without parameters shows as header line the drive and if necessary the name of a directory. E.g. The command "dir mpc" creates the header "Directory of C:\mpc\".

Below this, one line per directory or file is displayed with the information described below.

For a directory:

Date and time when the directory was created, identification “<DIR>“, directory name plus extension.

For a file:

Date and time of the last change of the file contents, file size in bBytes, file name plus extension.

The footer displays the total number of files and their total size in bytes as well as the number of directories.

Syntax

```
DIR [drive:] [/W] [/S] [/X] [/P] [path | file name]
```

/W - Displays only the names in wide format, with up to five file names and directory names in each line. Directory names are enclosed in square brackets.

/S - The output corresponds to the format of DIR without options in the Rtk operating system.

/X - Extended display. The footer additionally displays:

Total: - Disk size in bytes - number of clusters on the disk

Free: - Number of free bytes – number of free clusters

SectorsPerClusters: - Number of sectors per cluster

BytesPerSector: - Number of bytes per sector

/P - Pause after each full screen page. This option is not effective for remote CLI.

path | file name - The paths and file names to be displayed can contain the wildcards (*, ?).

1.4.2.9 DRIVES

Displays a list of available drives.

Syntax

```
DRIVES
```

Tries to mount all drives and lists all drives, which were successfully mounted. Floppy-drives without inserted disk are not listed, because they cannot be mounted.

```
DRIVES RAW
```

Lists all possible drives, even if they are not physically present and displays their current mount-state, without changing it.

```
DRIVES PHYSICAL
```

Tries to mount all drives and lists all drives including floppies that are physically available.

Requirements

LasalOS: Requires LasalOS 5.50 or later.

1.4.2.10 FC

Compares the contents of two files and displays the differences between them.

Syntax

```
FC filename1 filename2
```

filename1 - Name of file to be compared

filename2 - Name of the file to compare with filename1 or directory to look for a file with the same name as filename1

1.4.2.11 FORMAT

Formats a logical drive.

Syntax

```
FORMAT Drive:
```

Example

```
format a:
```

The FORMAT command creates a new root directory and file allocation table for the disk. It can also check for bad areas on the disk, and it can delete all data on the disk. In order for LASALOS to be able to use a new disk, you must first use this command to format the disk.

1.4.2.12 LABEL

Creates, changes, or deletes a disk volume label of a disk.

Syntax

```
LABEL drive: label
```

If no label is specified, any currently existing label is removed.

1.4.2.13 MD / MKDIR

Creates a new directory.

Short form: MD

Syntax

```
MKDIR pathname
```

pathname - Name of directory to create

Example

Suppose you want to create a directory on the disk in the current drive. To create a directory named MYDIR, type the following command.

```
mkdir \mydir
```

You could also type this command with the same results.

```
md mydir
```

Now suppose that the MYDIR directory is the current directory and that you want to create a subdirectory of MYDIR named PROJECTS. To create the PROJECTS directory, type the following command.

```
mkdir projects
```

1.4.2.14 MOUNT

Installs a Windows or Linux Samba share as a drive. Local directories can also be installed as drives. This command is only available in Salamander OS.

Syntax

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

type – Type of the mount. SAMBA, SAMBA_V1, SAMBA_V2 or LOCAL. With the types SAMBA_V1 and SAMBA_V2 the respective Samba protocol version is set. If <type> SAMBA is specified, Samba version 1 is used. You can also specify samba_string here. Then various settings (Samba protocol version, username, password etc.) can be made with a Linux option string. See -o.

netpath – Network path of the release. (IP address, DNS name or NETBIOS name \release) or local directory.

local drive – the drive where the network drive data are made available

-u – the user name of the mount path (optional).

-p – the password of the mount path (optional).

-o – Linux option string (-o and -u, -p are mutually exclusive). Here you can specify a lot of Linux options for Samba (e.g. sec, vers, username, password, domain etc.). For the operating systems Gecko and Salamander 'sec=ntlmv2' must be given. The string must not contain spaces. The individual options must be separated with a comma. See example. -o can only be used together with <type> = samba_string.

-RO – the network drive is installed readonly (optional and only for network drives)

-SYNC – Write accesses are not buffered but processed immediately (optional and only for network drives)

-t – Timeout time in ms for the resolution of NETBIOS names. The timeout time is only used to resolve NETBIOS names. The installation itself is not included in this time (optional, default = 2s).

Example

The following command mounts the DSWWXX/MountTest release as drive D:.

The timeout interval for the name resolution is set to 2 s.

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

The following command installs the DSWWZZ/MountTest share as drive E:.

The Samba protocol version 2.0 is used. The timeout time for the name resolution is set to 2 s. Write accesses are not buffered.

```
MOUNT samba_string \\DSWWZZ\MountTest E: -overs=2.0,username=MountUser,password=MountUser -S
```

```
MOUNT samba_string //123.123.123.123/SHARENAME D: -osec=ntlm,vers=2.0,username=user,password=
```

For the operating systems Gecko and Salamander4 'sec=ntlmv2' must be given.

```
MOUNT samba_string //123.123.123.123/SHARENAME D: -osec=ntlmv2,vers=2.0,username=user,password=
```

1.4.2.15 PARTITION

Creates a single partition spanning the whole disk-capacity. The partition is set active. All data may be destroyed.

Syntax

```
PARTITION pdi
```

pdi – physical disk index; this value for the desired disk can be found with command "DRIVES PHYSICAL" in column "PDI" of the result list. Valid values are in the range from 0 to 25.

Requirements

LasalOS: Requires LasalOS 5.50 or later.

1.4.2.16 RD / RMDIR

Removes a directory.

Short form: RD

Syntax

```
RMDIR pathname
```

pathname - Name of directory to remove

Example

To delete a directory named \USER\SMITH first ensure that the directory is empty like in the following example.

```
dir \user\smith
```

LASALOS should display only the "." and ".." symbols.

Then, from any directory except \USER\SMITH, type the following command.

```
rmdir \user\smith
```

You can type the following command with the same result.

```
rd \user\smith
```

1.4.2.17 REN

Changes the name of the file or files you specify.

Syntax

```
REN filename newname
```

filename - Name of file or directory to rename

newname - New name for the given file or directory

Example

To rename a file named CHAP10 to PART10, type the following command.

```
ren b:chap10 part10
```

The newly renamed file PART10 remains on drive B.

1.4.2.18 SMBSVR

With the package SAMBA-Server other controls can be granted access to the own file system. Also see command MOUNT. The SMBSVR command is used to manage users and file shares.

Syntax

SMBSVR START

starts the SMB server

SMBSVR STOP

stops the SMB server

SMBSVR USER ADD username password r/w path

configures a SMB user

username - name of the SMB user

password - password of the SMB user

r/w - general write or read access

path - home directory of the SMB user

With older Salamander versions (≤ 09.03.120) the user directory is also deleted when the Samba user is deleted. You should therefore not specify C:\ as user directory in the parameter <path>.



SMBSVR USER REMOVE username

deletes a SMB user

username - name of the SMB user

SMBSVR ACCESS ADD username/any sharename path r/w

configures a SMB share

username - name of the SMB user; if any is entered, you can connect to any user defined before

sharename - name of the share

path - directory to be shared

r/w - write or read access

Creating the share can last up to 90 seconds. If it should be executed immediately, the SMB server has to be stopped and restarted.

SMBSVR ACCESS REMOVE sharename

Deletes a SMB share

sharename - name of the share

Deleting the share can last up to 90 seconds. If it should be executed immediately, the SMB server has to be stopped and restarted.

Example

```
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

Displays a complete directory tree hierarchy.

Syntax

```
TREE pathname
```

pathname - Name of the directory to start the display with

1.4.2.20 TYPE

Displays the contents of a text file. Use the TYPE command to view a text file without modifying it.

Syntax

```
TYPE [drive:][path]filename
```

filename - Name of file to display.

Example

If you want to display the contents of a file named HOLIDAY.MAR, type the following command.

```
type holiday.mar
```



Displaying a file that is not a text file (e.g. *.BIN) is not recommended and may lead to unexpected behavior.

1.4.2.21 UMOUNT

UMount of a mounted network drive or local directory. This command is only available under Salamander OS.

Syntax

```
UMOUNT local drive
```

local drive – The drive where the network drive data are made available

Example

The following command removes the release, provided on drive D:.

```
umount d:
```

1.4.2.22 UNZIP

The command is used to extract (unzip) the zip file.

Short form: UNZIP

The command requires the DLM lslzip.dlm.



Syntax

UNZIP <in-file>	... file name /-path of the Zip file
<dest-dir>	... file name /-path of the unpacked target file

Example

The zip file "zipFile.zip" from the folder "C:\ZipFolder" should be unpacked as file "dest.txt".

1.4.2.23 VOL

Displays information about a volume.

Syntax

VOL [pathname:]

pathname - Name of a drive to display information about.

Only the drive letter is taken from the path.

1.4.2.24 XCOPY

Copies files and supports wildcards.

Syntax

XCOPY source destination

source - Files to be copied (*.ext, *.*).

destination - File name and/or directory of the new files.

Remarks

The LasalOS tries to allocate the destination file in adjoining disk sectors to reduce disk fragmentation and improve file access speed. If not possible, the text message "Information: file not contiguous" is displayed. Although this is neither an error nor warning message, it is an indication that the disk is either heavily fragmented or rather full.

This is a real error message, which means that the destination file could not be created and indicates that the destination disk needs to be reformatted.

Both topics are described in the LasalOS User Guide, chapter "How to setup/defragment" a Compact flash medium".

1.4.2.25 ZIP

The command is used for compressing (packing/zipping) files. Only one file can be compressed. If an existing Zip file is defined as the destination, the existing will be overwritten by the new one.

The command requires the DLM `lslzip.dlm`.



Syntax

```
ZIP      <in-file>          ... File or path for the file(s) to archive
          <out-file>        ... File or path for the ZIP file to be created
          <pack-level>       ... Compression level (1 = fastest, .., 9 = largest compression)
```

Example

The file `source.txt` from the folder "C:\Source Folder\" is compressed into a zip file named "zipFile.zip" in the current folder.

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

1.4.3 Operating System Commands

1.4.3.1 ADDR

Displays the address of a linked variable.

Syntax

```
ADDR variable
```

Example

```
addr OPS
```

To display the address of the variable OPS:

1.4.3.2 APPTASKS

Displays all Application Real-Time, Cyclic and Background tasks.

Syntax

APPTASKS

1.4.3.3 ARP

Displays the physical MAC addresses of connected TCP clients.

Syntax

ARP

1.4.3.4 BGTASKS

Displays all application background tasks.

Syntax

BGTASKS

1.4.3.5 BIOS

Displays BIOS information.

Syntax

BIOS

1.4.3.6 BOOT

Installs a new operating system on the hard drive. The following note is only valid for Salamander operating systems. If the version number of the currently installed Salamander operating system is in the range of 09.03.102 to 09.03.143, problems with installed SW packages may occur. There are two possibilities to avoid these problems:

Possibility 1

All installed SW packages must be uninstalled with PACKAGE UNINSTALL before a new Salamander operating system is installed. After installing the new Salamander operating system,

the SW packages must be reinstalled with PACKAGE INSTALL. For details see the CLI command [PACKAGE](#).

Possibility 2 The new operating system must be installed twice in immediate succession.

Syntax

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

image - File name of the .LBI or .RTB image to install.

dest-drive - Drive, one which the new image is installed. If not specified, the image is installed on drive "C".

Remarks

Starting from OS version 01.01.096, a new boot image file format can be used. The LBI (Lasal Boot Image) format contains the operating system image and security mechanisms, such as an ID for the target platform and a CRC32 to avoid installing a corrupt OS image.

 Starting from LasalOS 01.02.096, LBI files should be used only!

Starting from LasalOS 01.02.150 only LBI files are supported.

Example

```
BOOT c:\etvedge.lbi
```

Return values

The following error codes can be returned when the command is used via a rexx script:

Value	Description
-1	Internal error, cannot determine PLC type
-2	Invalid Parameter (no filename given, invalid drive, ...)
-3	File not found
-4	LBI not enough memory
-5	LBI wrong CRC

-6	LBI invalid header
-7	LBI invalid file length
-8	LBI invalid version
-9	LBI contains an invalid image for the CPU
-10	LBI contains no RTB image file
-11	LBI error opening file
-12	LBI error write file
-13	Not enough memory
-14	Internal error, create full filename
-1001	Error install new image, not enough memory
-1002	Error install new image, invalid RTB file
-1003	Error install new image, error device
-1004	Error install new image, no valid boot code
-1005	Error install new image, creating boot image file, can also returned if the disk is full
-1006	Error install new image, not contiguous
-1007	Error install new image, invalid sector size
-1008	Error install new image, writing boot sector
-1009	Error install new image, invalid RTB file
-1010	Error install new image, boot code corrupted
-1011	Error install new image, error opening RTB-file
-1012	Error install new image, unsupported
-1013	Error install new image, not supported
-1014	Error install new image, buffer to small
-1015	Error install new image, disk full, -1005 can also returned, depends on fragmentation

Requirements

LasalOS: Parameter 'dest-drive' requires LasalOS 5.50 or later. Earlier releases always install the new image on drive 'C:'.

1.4.3.7 BOOTPIC

Set options on how the bootpic is displayed

Syntax

BOOTPIC (option) (option) ...
enable displaying bootpic

Options

SHOWCLI	disable "Press TAB to show the CLI"
AUTOEXEC	disable "Autoexec.lsl finished"
QUITONCLISTART	disable bootpic after CLI has started
NOT	inverts all set options

Example

BOOTPIC NOT QUITONCLISTART

The bootpic is enabled, everything is off and the bootpic has to be manually disabled

1.4.3.8 BOXES

Displays information about all mailboxes.

Syntax

BOXES

1.4.3.9 BUFFERS / BUF / B

Displays information about the files buffer cache.

Short forms: BUF and B

Syntax

BUFFERS

1.4.3.10 CANINFO

Displays information about the CAN settings.

Syntax

CANINFO ifnum

ifnum – Number of the CAN interface (1 = CAN1, 2 = CAN2, ...).

1.4.3.11 CIL

Displays all registered common interface libraries.

Syntax

CIL

1.4.3.12 CYTASKS

Displays all application cyclic tasks.

Syntax

CYTAKS

1.4.3.13 DATE

Shows current date and time.

Syntax

DATE

1.4.3.14 DUMP

Dump application memory.

Syntax

DUMP Start [End]

Start - Start address.

End - End address.

When the end address is not specified, a default value of Start + 15 is assumed. The address values must be specified in a hexadecimal format.

There must be a valid workspace and the address range must lie in the User-Code or User-Data section of RAM.

Example

```
DUMP f002ec
```

Dump of 16 bytes starting at address F002ECh

```
DUMP f002ec f002ef
```

Dump from address F002ECh through address F002EFh

1.4.3.15 ERRORLOG

A shortcut to inspect the kernel error logs.

This command will actually store the Logs to a "TEMPERR.LOG" file and then display it page-wise. Afterwards, the temp file is deleted.

Syntax

```
ERRORLOG
```

1.4.3.16 EXCEPT

Displays or resets exception information.

Syntax

```
EXCEPT
```

Displays exception information.

```
EXCEPT RESET
```

Resets exception information.

1.4.3.17 FCLOSEALL

Writes all cached buffers to disk and closes all currently open files.

Syntax

FCLOSEALL

1.4.3.18 FILE

LASAL Data File Commands (active.dat).

Syntax

FILE LOAD

Loads a data file.

FILE LOAD ACTIVE

Loads the default active data file.

FILE LOAD ACTIVE filename

Loads the specified file as the ACTIVE data file.

FILE LOAD mask filename

Loads the specified file using the specified HEX mask.

FILE SAVE

Saves a data file.

FILE SAVE ACTIVE

Saves the default active data file.

FILE SAVE ACTIVE filename

Saves the specified file as the ACTIVE data file.

FILE SAVE mask filename

Saves the specified file using the specified HEX mask.

FILE DISPLAY

Displays a data file.

FILE DISPLAY filename

Displays the specified data file.

1.4.3.19 FILES

Displays information about all currently open files.

Syntax

FILES

Example

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

The column Index contains the file slot that the file occupies in the internal file table; Handle is the hexadecimal representation of the file's handle. The Flags column may shows a combination of the letters R, W, D, and S that represents read only access, read/write access, directory, and shared access respectively. FilePos gives the current value of the file's pointer; Name is the full path name of the file. Use this command to analyze which files are open if a program does not properly close all files.

1.4.3.20 FLUSHLOG

Flushes the contents of the operating system buffer to a file.

Syntax

FLUSHLOG

1.4.3.21 GET ERRORLOG

Store logged Kernel errors.

This command stores the logged Kernel errors to a specified file. If no filename is specified, it creates "ERRORLOG.TXT".

Syntax

GET ERRORLOG [file name]

1.4.3.22 HANDLES

Displays a list of all currently existing handles.



This function is only available under the operating system RTK. It is thought for internal use only!

Syntax

HANDLES

Example

```
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

Displays information about the applications Heap.

Syntax

HEAP

Example

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

1.4.3.24 HELP / ?

Displays help information.

Short form: ?

Syntax

```
HELP
```

Displays all CLI commands.

```
HELP command
```

Displays details about command 'command'.

1.4.3.25 INFO

Displays information to some specific topics. Serves mainly diagnostic purposes.

Syntax Operating System RTK

```
INFO USB
```

Shows the USB host and connected devices.

```
INFO INFOBLOCK
```

Shows the Sigmatek device info block.

```
INFO XREGS
```

Shows the Xilinx register.

```
INFO SIGMAIRQ
```

Shows the Sigmatek device interrupts.

Syntax Operating System Salamander

```
INFO SERNUM
```

Shows the serial numbers of various devices (e.g. CPU and SD card).

```
INFO VER
```

Shows the version numbers of operating system and Hardware (FPGA).

INFO MEM

Shows the current memory usage.

INFO MEMREQUIRE

Shows the memory requirements of the operating system and the installed SW packages.

INFO CONN

Shows information about interfaces and connections.

INFO SDCARD

Shows information about the SD card.

INFO CPUTEMP

Shows the CPU temperature(s).

1.4.3.26 INTS

Displays information about interrupts since program start.

This command is only available with the RTK operating system.



Syntax

INTS

For each IRQ registered by the operating system where interrupts have occurred since the program start, a single line of information is output. The IRQ number, the number of interrupts that occurred, the unused interrupt stack and the number of recursive interrupts (normally 0) are displayed. The cumulative CPU time for each IRQ is returned only in the debug version of the operating system.

If recursions have occurred (as with the keyboard interrupt in the example), statistics for the panic stack are also given. Recursions on the panic stack (column "Doubles" in row "Panic") mean danger and indicate an error in the interrupt handler or an interrupt overload.

Example

INTS

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

1.4.3.27 LINK

Displays linker information.

Syntax

LINK INFO

Displays general linker information.

LINK MODULES

Displays information about all linked modules of this project.

LINK GLOBALS

Displays information about global symbols of this project.

1.4.3.28 LISTTS

Displays information about the stack area of the operating system tasks.

Syntax

LISTTS

The first columns show the name of the task, then a comma separated list of the following values is displayed:

- start- and end-address of the stack
- the current stack-pointer (ESP)
- the least number of bytes that have been free on a task's stack since it was created

Example

Main Task	:00D00000-00DBFF88,00DBFE84,783148d
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

Enables or Disables logging features.

Syntax

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

Shows state of ACTIVE.DAT logging.

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

Enables/Disables writing an ACTIVE.LOG-file while writing the ACTIVE.DAT at Reset/Power down.

1.4.3.30 MEM

Displays information about the operating systems heap memory.

Syntax

```
MEM
```

1.4.3.31 MEMDUMP

Displays the content of a memory area.

Syntax

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

When the optional parameter filename is specified, then the content of the memory is copied to a file in a binary format.

Example

```
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 E8 03 00 00 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

Displays information about the current TCP connections.

Syntax

NETSTAT

Example

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

Prot	The connection's protocol
Local Host	The IP address of the local machine
Remote Host	The IP address of the remote client
Status	Can be READY, BUSY or ERROR

1.4.3.33 NETWORK

Displays information about installed Ethernet devices.

Syntax

NETWORK

Displays information about all installed Ethernet devices.

NETWORK iface

Displays information about Ethernet interface with number 'iface'.

1.4.3.34 NUMLOCK

Displays or alters current keyboard numlock status.

Syntax

NUMLOCK

Displays current numlock status.

NUMLOCK ON

Sets numlock on.

NUMLOCK OFF

Sets numlock off.

Requirements

LasalOS: Requires LasalOS 5.50 or later.

1.4.3.35 PACKAGE

With this CLI command, such additional functions as a file server can be later installed.

For installation, a SIGMATEK installation packet (ZIP archive) of the desired function is required. Unpack the contents of the archive into the C:\lslsys\packages directory. Next, the SIGMATEK installation packet can be installed with the command "PACKAGE INSTALL <Name of Packet>".

A readme.txt file can be found in the SIGMATEK installation packet, which describes the installation process of the respective packet.



Syntax

PACKAGE [Option] "Command" "Packet name"

Options:

-l - Delayed installation of the packet (after next reboot)

-d - Shows the debug output in the CLI

Commands:

LIST - Shows a list of the installation packet

INSTALL - Installs the specified packet

UNINSTALL - Removes the specified packet

Examples

PACKAGE LIST

Shows the list of all installed packets

PACKAGE INSTALL samba

Installs the Samba packet

PACKAGE -l INSTALL samba

Installs the Samba packet after the next reboot

Requirements

Lasal OS Salamander ab Version 09.03.030.

1.4.3.36 PING

Sends a ping signal to a remote host and checks the answer. 4 packets of 32 byte data are transmitted.

Syntax

```
PING ip address
```

The IP address of the remote host (must be specified in dotted decimal notation).

A ping can be used to determine if a machine can be reached on the network.

Example

```
PING 192.168.44.106
```

1.4.3.37 PNPBIOS

Displays info about PnP and PCI bios.

Syntax

```
PNPBIOS
```

1.4.3.38 QUCMD

Puts a command string into a queue that is executed after the execution of the autoexec.lsl. This command has no effect when it is called outside the autoexec.lsl.

Syntax

```
QUCMD command string
```

1.4.3.39 REBOOT

Resets and reboots the machine.

Syntax

REBOOT

1.4.3.40 REXX

Executes a REXX program.

Syntax

REXX [program-name]

program-name is the name of the Rexx program to be executed. If no program name is specified, LasalOS-Rexx waits for Rexx commands to be typed in and will execute those commands when the end-of-file character (Ctrl-Z) is typed.

Remarks

See documentation [LasalOS Rexx](#) for a description of REXX.

1.4.3.41 ROUTE

Displays and manipulates the routing table.

Syntax

PRINT

Displays the entries in the routing table.

ADD

Adds an entry to the routing table.

DEL

Deletes an entry from the routing table.

The routing table consists of 5 columns. The first column contains the network destination address, the second the network mask belong to it. The third entry is the gateway address – this is the destination address of this route. The fourth is the network interface (in IP-Address format) and the last one is metric.

To add an entry in the routing table these 5 entries must be set. To delete an entry only the network destination address and the network mask must be given.

1.4.3.42 RTTASKS

Displays all Application Real-Time Tasks.

Syntax

RTTASKS

1.4.3.43 SEMAS

Displays a list of currently existing semaphores.

Syntax

SEMAS

SEMAS will display the semaphore's names, types, values, and a list of all tasks waiting at the respective semaphore. For resource and mutex semaphores the name of the owning task (if any) is also given.

1.4.3.44 SET CYCLIC

Displays or sets the time slot for low priority tasks.

You should only change the Cyclic task delay time, if you really know what you are doing!

Syntax

SET CYCLIC

Displays the Cyclic task time slot.

SET CYCLIC DELAY value

Sets the Cyclic task time slot (value: Cyclic task time slot in ms).

1.4.3.45 SET DBGHEAP

Activates/deactivates the debug heap settings. These settings only become effective after a reset run.

These settings are for debugging purposes only. They can help finding memory errors like

- Realloc or Free with an invalid pointer or a pointer to a memory block that was already freed
- Not enough memory available (if this error is not handled by the application!)
- An application using the old pointer after Realloc() instead of the one returned by Realloc()
- Writing before or after an allocated block of memory
- Writing to block of memory which was already freed

Since the application heap is checked during each heap operation (Malloc, Realloc, Free,) for errors, heap operations can be quite time-consuming. If the application uses many heap operations, the application may slow down considerably and possibly result in runtime-errors. When using visualization especially, changing pages/screens may be slow.

If the heap is used extensively, it is recommended that SET DBGHEAP ALL ON be used while developing an application. If development is complete and it is certain that no further memory errors exist, the heap check should be disabled for production.

Syntax

SET DBGHEAP

Displays the current DBGHEAP settings.

SET DBGHEAP CHECK_ALWAYS ON|OFF

ON - The entire application heap is checked for integrity on every heap function call.

OFF - Only the block specified in ReAlloc() or Free() is checked according to the other DBGHEAP settings.

SET DBGHEAP ERR_OUTOFCMEM ON|OFF

ON - The OS raises an AppMem error when Malloc() or ReAlloc() fails because of a memory shortage.

SET DBGHEAP REALLOC_NEW_PTR ON|OFF

ON - The ReAlloc() function always allocates a new block at a different memory address than that of the original block.

SET DBGHEAP CHECK_LIMIT_BEGIN ON|OFF

ON - The heap manager allocates a larger block than requested for a small buffer on the beginning side of your data to catch overwrites of the allocated region before the start of

the region. A 4 byte buffer zone before the data area is filled with 16#FD under RTOS and 16#FC under Salamander. The OS triggers an AppMem error when releasing this block with Free() or with every heap function call (see CHECK_ALWAYS ON|OFF) if the bit pattern in the buffer zone has been overwritten.

SET DBGHEAP CHECK_LIMIT_END ON|OFF

ON - The heap manager allocates a larger block than requested for a small buffer on the ending side of your data to catch overwrites of the allocated region after the end of the region. A 4 byte buffer zone before the data area is filled with 16#FD under RTOS and 16#FC under Salamander. The OS triggers an AppMem error when releasing this block with Free() or with every heap function call (see CHECK_ALWAYS ON|OFF) if the bit pattern in the buffer zone has been overwritten.

SET DBGHEAP FILL_FREE_BLOCK ON|OFF

ON - The heap manager fills freed blocks with (0xDD). So the application can check whether freed memory is still being written to.

SET DBGHEAP FILL_NEW_BLOCK ON|OFF

ON - New blocks are filled with 0x0CD when they are allocated.

SET DBGHEAP DBGHEAP STORE_IP ON|OFF

ON - The heap manager allocates a larger block than requested for a buffer that holds the IP (instruction pointer) address where the heap function was called. In the case of an AppMem error the IP address of the block in error is written to the system log.

SET DBGHEAP CHECK_FREE_PTR ON|OFF

ON - The OS triggers an AppMem error when Free() specifies a memory block that was already freed.

SET DBGHEAP ALL ON|OFF

ON - Activates all of the above debug heap settings.

OFF - Deactivates all of the above debug heap settings.

SET DBGHEAP TRIGGERADDR <addr>

If <addr> is not zero, the functions MALLOC and REALLOC will write the call-stack to the event-log file. This is useful e.g. if you know the block-address of a corrupted memory block and you want to know the function who allocated it. You can further examine the code and investigate what happens with the memory.

Requirements

The debug heap functions are available since LasalOS 5.52.

The setting TRIGGERADDR is available since LasalOS 1.1.51

1.4.3.46 SET DEBUG

Tells LasalOS, which debugger will be used.

Syntax

```
SET DEBUG LASAL
```

Lasal Class debugger for debugging applications (default).

```
SET DEBUG MSDEV|RTD32
```

Microsoft VC or Borland debugger is used.

1.4.3.47 SET ERRORLOG

Displays or enables/disables kernel error logging.



Since LasalOS 5.42 this command is obsolete. Use [SET EVENTLOG](#) instead.

Syntax

```
SET ERRORLOG
```

Displays the settings for kernel error logging.

```
SET ERRORLOG ON|OFF
```

ON enables, OFF disables kernel error logging.

If kernel error logging is enabled, use "[GET ERRORLOG...](#)" to inspect the log file.

1.4.3.48 SET EVENTLOG

Displays or sets the parameters for logging of operating system messages.

Syntax

```
SET EVENTLOG
```

Displays the settings for logging of operating system messages.

```
SET EVENTLOG ON [file-quota]
```

Enables logging of operating system messages to a message buffer. Parameter file-quota can be used to enable or disable writing the operating system message buffer to a disk file and to specify the maximum size of the disk file. file-quota can be set to one of the following values, the default value is -1:

Value	Meaning
0	Disables writing the operating system message buffer to a log-file.
-1	Enables writing the operating system message buffer to a log-file, a system default value is used for the maximum size of the log-file.
Any other value > 0	Enables writing the operating system message buffer to a log-file, a system default value is used for the maximum size of the log-file.

The name of the operating system message file is EVENT00.LOG. The default path for this file is C:\SYSMSG. This path can be overridden by setting the environment variable SYSMSGPATH to another value. To set environment variable, use the CLI command [SETENV\(\)](#).

See the [LasalOS Users Guide](#) for a description of considerations for writing operating system messages to a message file (chapter sysmsg facility).

SET EVENTLOG OFF

Disables logging of operating system messages to a message buffer.

Remarks

The default setting is SET EVENTLOG OFF.

Requirements

LasalOS: Requires LasalOS 5.42 or later.

1.4.3.49 SET SYSTRACE

Enables or disables the PLC trace facility.

Syntax

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

start-condition und stop-condition are bit masks, where one bit specifies a certain event that should trigger the start or stop condition for the trace. A bit wise OR combination of the following values is possible (the prefix 0x means hexadecimal format):

Value	Meaning
0x00000001	Run
0x00000002	Reset
0x00000004	Error (e.g. Access Exception)

0x00000008	CDias Watchdog
0x00000010	Runtime Error
0x00000020	Keyboard or mouse events
0x00000040	Reserved for LSE kernel
0x00000080	Reserved for LSE kernel
0x00000100	Reserved for application events
0x00000200	Reserved for application event
0x1<n>000000	Event after n seconds
0x40000000	Immediate
0x80000000	Trace buffer full

type-mask is a bit-mask, where one bit specifies the type of trace entry that should be written into the trace buffer. A bit wise OR combination of the following values is possible (the prefix 0x means hexadecimal format):

Value	Meaning
0x0001	Task switches
0x0002	Change of application state (Run, Reset, Error)
0x0004	Dias events
0x0008	Calls of cyclic application methods (rtwor, cywork, background)
0x0010	Reserved for interrupt events
0x0020	Reserved for CAN events
0x0040	Keyboard or mouse events
0x0080	LSE kernel event types 0
0x0100	LSE kernel event types 1
0x0200	User event types 0
0x0400	User event types 0

When the start and stop conditions have been met, the trace buffer is written to the file C:\SYMSMG\TRACE00.CSV. This file can be viewed with the Lasal2 PLC-Trace-View. After the trace buffer has been written to a file, the start condition is reset and must be re-enabled when another trace should be activated.

The parameters can be entered in decimal or in hexadecimal format. The prefix 0x specifies hexadecimal format.

The size of the trace buffer can be set with the SET TRACEBUFSIZE command.

Note that a running trace may affect runtime behavior, so be careful when activating the trace in a production environment.

Examples

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

Trace starts at RUN and ends when a runtime error occurs. All events are written to the trace buffer.

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

Trace start immediate and ends when the buffer is full. Only events of the type task switch are written to the trace buffer.

Requirements

LasalOS: Requires LasalOS 01.01.086 or later.

1.4.3.50 SET TRACEBUFSIZE

Sets the size of the buffer in bytes that is used for the PLC trace facility.

Syntax

```
SET TRACEBUFSIZE size
```

Operating system Salamander: TRACEBUFSIZE can only be set in AUTOEXEC.LSL. The size must be set between 128 kB and 3 MB.

1.4.3.51 SET USB_CACHE_FLUSHING

Activates or deactivates the explicit flushing of the writing cache on a connected USB stick via additional USB transactions or shows the current operating status of this mechanism.

USB sticks have an internal, volatile memory, where data receives via USB are stored, before they are written in the non-volatile flash memory by the firmware of the USB stick. If the USB stick is disconnected from the USB and with that from the power supply, before the contents of this cache is completely written to the flash, data is lost. Under Lasal OS it should be possible for a user to unplug an USB stick without before informing the OS

about unplugging resp. "ejecting" the stick - like e.g. under some desktop operating systems. So Lasal OS as a default triggers a transfer of the data in the cache of an USB stick to the flash after certain file operations by sending a certain command to the firmware of the stick. Using this command usually leads to no resp. insignificant losses in the reached data rate for writing access to the USB stick. For the case that in future too low data rates are measured for a certain USB stick model, the described mechanism can be deactivated with the command SET USB_CACHE_FLUSHING OFF. Because of this risk of a data loss - depending on the model of the USB stick - this should only be used temporary resp. for testing purposes.

The default setting is SET USB_CACHE_FLUSHING ON.



Syntax

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

Shows the current operating status of the cache flushing mechanism.

```
SET USB_CACHE_FLUSHING ON or SET USB_CACHE_FLUSHING 1
```

Activates explicit flushing of the writing cache of the connected USB sticks.

```
SET USB_CACHE_FLUSHING OFF or SET USB_CACHE_FLUSHING 0
```

Deactivates explicit flushing of the writing cache of the connected USB sticks.

```
SET USB_CACHE_FLUSHING HELP
```

Shows information about the usage and effects of this command.

Requirements and Availability

The command SET USB_CACHE_FLUSHING is available for the platforms C-IPC, ETV and ETV-EDGE since Lasal OS version 01.03.070.

1.4.3.52 SETENV

Sets or deletes the value of an environment variable.

Use command [SHOWENV](#) to display the value of an environment variable.



Syntax

SETENV variable name

This command deletes the environment variable.

SETENV variable name value

This command sets the value of an environment variable.

Requirements

LasalOS: Requires LasalOS 5.42 or later.

1.4.3.53 SETLEDSTATUS

Sets a LED status.

Syntax

SETLEDSTATUS led status

led - RUN, STATUS, ERROR

status - OFF, ON, FLASH

Requirements

LasalOS: Starting with Salamander 09.01.140.

1.4.3.54 SHOWENV

Displays the value of an environment variable.

Use command [SETENV](#) to set or delete the value of an environment variable.



Syntax

SHOWENV variable name

Requirements

LasalOS: Requires LasalOS 5.42 or later.

1.4.3.55 SRAMLOAD

Restores the SRAM from a file, which was previously created with SRAMSAVE.

Syntax

```
SRAMLOAD filename
```

Requirements

LasalOS: Requires LasalOS 1.1.8 or later.

1.4.3.56 SRAMSAVE

Saves the contents of the SRAM to a file.

Syntax

```
SRAMSAVE file name
```

Requirements

LasalOS: Requires LasalOS 1.1.8 or later.

1.4.3.57 STATUS

Displays LASAL OS Status Information.

Syntax

```
STATUS
```

1.4.3.58 TASKS1 / TASKS

Displays information about the currently running tasks.

Short form: TASKS

Syntax

TASKS1

1.4.3.59 TASKS2

Displays information about the currently running tasks and their waiting positions.

Syntax

TASKS2

1.4.3.60 UNMOUNT

Properly remove a USB stick.

Syntax

UNMOUNT <device letter>

Example

UNMOUNT E

Unmounts the Device E:\

1.4.3.61 VER

Displays the LASAL OS version with release-date, current resolution and contact info. Additionally this command can also be used to display the version of a DLM. For that an optional transfer parameter ddlmname.dlm has to be assigned.

Syntax

VER

VER -D zlib.dlm

1.4.3.62 vpndaemon

The VPN client service can be started or stopped manually with the (remote) CLI command. The optional parameter <configuration> is the name of the VPN configuration file without the extension .conf. The VPN configuration files must be in the folder C:\lslsys\vpn. The name of a VPN configuration file must have the extension .conf.

The command is only available on Edge2 CPUs with Salamander OS starting with version 09.03.100.



Syntax

```
vpndaemon start [<configuration>] | stop [<configuration>] | restart [<configuration>] | reload
```

Examples

```
vpndaemon start
```

Starts a VPN daemon for each configuration file in the folder “C:\lslsys\vpn”.

```
vpndaemon start <configuration>
```

Starting a VPN daemon with the VPN configuration file <configuration>.conf. The VPN configuration file must be in the folder “C:\lslsys\vpn”.

```
vpndaemon stop
```

Stops all running VPN daemons.

```
vpndaemon stop <configuration>
```

Searches for a running VPN daemon started with the specified VPN configuration file and stops it. The VPN configuration file must be in the folder “C:\lslsys\vpn”.

```
vpndaemon restart
```

Stops all running VPN daemons and starts a VPN daemon for each VPN configuration file in the folder “C:\lslsys\vpn”.

```
vpndaemon restart <configuration>
```

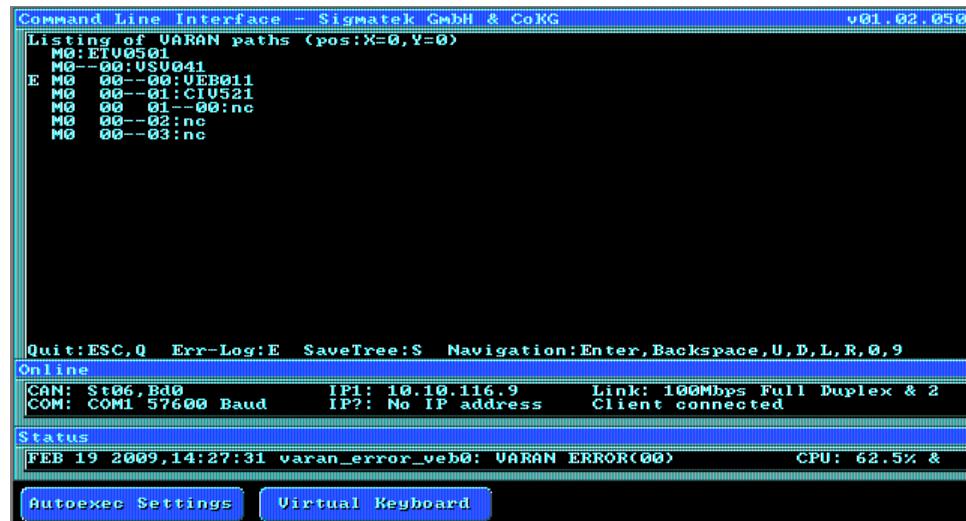
Searches for a running VPN daemon started with the specified VPN configuration file and stops and restarts it. The VPN configuration file must be in the folder “C:\lslsys\vpn”.

```
vpndaemon reload
```

Executes a restart for all running VPN daemons. If a VPN daemon does not run under the user root, a corresponding VPN configuration file must exist in the folder “C:\lslsys\vpn”.

1.4.3.63 VT

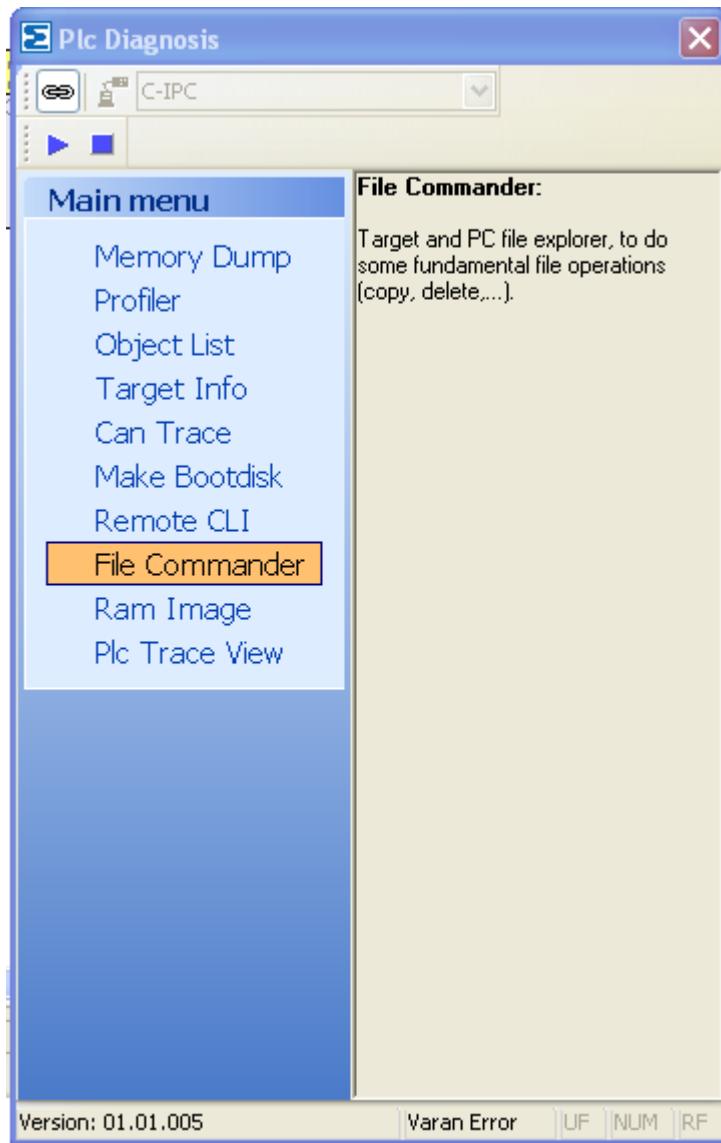
Through the display of the error tree with VT, the following screen is shown. The error is identified with E. Here, the SaveTree:S key can be used to save the entire tree in the C:\SYMSMG\ErrorTree.txt file.



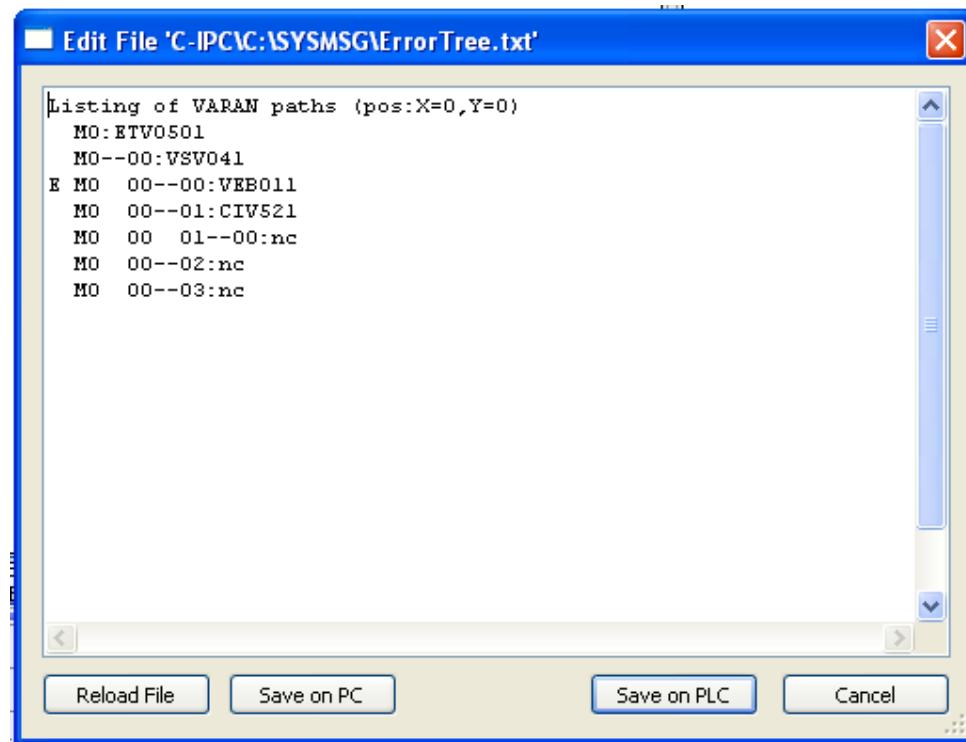
Listing of VARAN paths (pos:X=0,Y=0)

```
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
```

The file can be read with an online connection.



C:\SYSMSG\ErrorTree.txt



1.4.4 Projects Commands

1.4.4.1 SET FPUROUNDCONTROL

Changes the rounding mode of the FPU.

Only for users that have the corresponding knowledge for the FPU!



Syntax

```
set fproundcontrol x
```

Value from 0-3

- | | |
|---|------------------|
| 0 | Round to nearest |
| 1 | Round down |
| 2 | Round up |
| 3 | Truncate |

Requirements

LasalOS: Requires OS Version > 01.02.051

1.4.4.2 LSLLOAD / LO

Loads a saved project from disk.

Short form: LO

Syntax

LSLLOAD [drive:][path]

For easy usage, standard load path is c:\LSLWORK\PROJECT.IDX.



1.4.4.3 LSLSAVE / SA

Save the current project onto disk.

For easy usage, standard save path is c:\LSLWORK\PROJECT.IDX. If named, the index file is duplicated as "PROJECT.IDX", to use with LSLLOAD without a given path.

If the option NORESET is specified while the application is running, the project will be stored without stopping the application or triggering a reset. While saving the application, latencies can occur. This option can only be used with Lasal Class 2 (LC2) / PLC Diagnosis (Remote CLI) starting with versions 02.02.157 (LC 2) and 01.01.041 (PLC Diag). Older versions trigger a reset automatically.

For a negative return value see the list at the end of the document.

Short form: SA

Syntax

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

1.4.4.4 RUN

Attempts to RUN the LASAL application.

If RUN is executed within the autoexec.lsl, and there's a online communication, the application remains in the RESET mode.

If RUN is executed within the autoexec.lsl and the argument UC is added, an online communication is ignored and the application starts.

Starting the application from the Command Line Interface, RUN and RUN UC have the same behaviour.

Syntax

```
RUN  
RUN UC
```

1.4.4.5 RESET

Sets the LASAL OS into RESET mode.

Syntax

```
RESET
```

1.4.4.6 PROJECT

Displays project information.

Syntax

```
PROJECT command  
PROJECT MODULES
```

Displays project module list.

```
PROJECT LINK
```

Links the current project.

```
PROJECT ERRORS
```

Displays project linker errors.

```
PROJECT INFO
```

Displays project information.

PROJECT SYMBOL

Displays project symbol information.

PROJECT SAVE [directory]

Saves project to specified directory on HDD.

(directory defaults to C:\LSLWORK)

PROJECT LOAD [directory]

Loads project from specified directory on HDD.

(directory defaults to C:\LSLWORK)

PROJECT CLEAR

Removes actual project out of memory.

1.4.5 Configuration Commands

1.4.5.1 ADDBR

This command creates a bridge network interface. It is only available with the operating system Salamander starting with version 09.03.177.

With a bridge, several physical networks on several IP interfaces can be combined into a common network, which is then connected to a logical bridge network interface.

The combined networks have the same subnet mask. The control on which the bridge has been configured then operates similarly to a switch.

The bridge presents itself as IP50. You can assign an IP address to it so that the device on which the bridge is located can be addressed. The interfaces, which are put into the bridge, do not have an IP address anymore.

Syntax

ADDBR 50 <list of interfaces to be added> [stp]

If "stp" is specified, the spanning tree protocol is activated (default is not active).

Spanning tree protocol prevents unwanted circling packets.

Since the spanning tree protocol causes an increased CPU load, care should be taken to design the network topology in such a way that no loops are possible and that the spanning tree protocol can be omitted.

Example

Combination of a WLAN and a cable network

ADDBR 50 1 10 11

adds IP1, IP10 and IP11 to bridge IP50

Combination of two Ethernet interfaces

ADDR 50 1 2

adds IP1 and IP2 to bridge IP50

The bridge can then also be given an IP address

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

1.4.5.2 CALIB

Starts a calibration program for the touch.

Syntax

CALIB

For OS versions ≥ 1.1.214:

CALIB [nx ny]

nx – The number of calibration points in x-direction

ny – The number of calibration points in y-direction

Follow the instructions and the calibration data is saved to the TOUCHCFG.DAT file in the root directory.

Note

In order to use the virtual touch keyboard (e.g. in an C-IPC's CLI) a valid touch interface has to be configured, as well as placing the *.smi Files into the c:\lslsys directory. These files can be found on the Master-CF-card.

1.4.5.3 Calib check

This command is only available with the Salamander operating system. It must be entered in the autoexec.lsl file.

When the control starts up, this command checks whether the touch screen has already been calibrated. If the file C:\touchdat.cfg is not available, the calibration of the touch screen is started. See command [CALIB](#).

1.4.5.4 CHANGECERT

Syntax

CHANGECERT option <filename>

Options

-s / -svr	For changing server cert + key file
-c / -cli	For changing client cert

Examples

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

For server

CHANGECERT -cli c:/path/to/cert

For client

1.4.5.5 DIAS

Displays available DIAS stations.

Syntax

DIAS [nr]

nr - The number of a DIAS station

Additionally, if a DIAS MASTER is present, these functions become also available:

DIAS UPDATE filename sector

Updates the DIAS-MASTER program. Before updating, a backup BIN file of the flash content is created in the \LSLSYS\ system directory with a unique file name.

filename - The HEX(MCS) or BIN file to update.

sector - The number of the flash-sector to update (0 or 1).

DIAS BACKUP filename sector

Saves the actual content of a flash-sector to a BIN file, which can be used to restore old versions.

filename - The name of the backup BIN file.

sector - The number of the flash sector to backup (0 or 1).

DIAS VERIFY filename sector

Compares a specified file and the content of a flash sector.

filename - The HEX(MCS) or BIN file to compare.

sector - The number of the flash-sector to compare (0 or 1).

DIAS INFO

Displays the file names of the current MASTER-programs in both flash sectors.

1.4.5.6 EURO

Sets date/time display to European format.

Syntax

EURO

To switch back to the US format, use the [US](#) command.

1.4.5.7 FIREWALL

This command is used to lock and unlock ports. Important: Under no circumstances does it replace a professional upstream firewall. We also strongly advise against connecting a control to the Internet without a professional firewall.

Syntax

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

Commands

LIST	Lists the configured rules in iptables format
LISTLONG	Lists the configured rules in a detail and plain text format
BLOCKALL	Everything is blocked except the loopback device
SIGSTD	Everything is blocked except the loopback device and the SIGMATEK standard ports Open Input Ports (TCP) 1954 LASAL communication 1955 Comlink 1956 Comlink refresh list 2054 LASAL communication (SSL) 2055 Comlink (SSL) 2056 Comlink refresh list (SSL)

	2402 Data service loader
	9980 Data service
	Open Input Ports (UDP)
	1954 LASAL IP scanner
	Other Protocols
	ICMP (PING)
REMOVEALL	Removes all configured rules
REMOVEBYNUMBER	Removes a rule by chain and number
SAVE	Save the current configuration
LOAD	Load a saved configuration
STARTACCEPT	If you want the configure the ports by yourself, this command MUST be called ONCE after the command BLOCKALL. Do not use this command if you want to configure additional rules to the SIGMATEK standard configuration
ACCEPTPORT	Open a port for one or all Ethernet interfaces
ACCEPTADDRESS	Accept everything from a defined IP address for one or all Ethernet interfaces
NO COMMAND	Use the IPTABLES syntax

Examples

- The ports must be configured in the autoexec.lsl.
- The ports should be configured, before the ethernet interfaces will be activated (SET IP ...)
- The ports cannot be configured with the "ServiceProvider"
- !!! Do not make a rule inefficient by defining a counter rule, but delete this rule !!!

Example 1 (Sigmatek standard)

```
FIREWALL REMOVEALL
FIREWALL SIGSTD
```

Example 2 (Allow a VNC client to connect over all interfaces - additional to SIGSTD)

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

Example 3 (Allow a VNC client to connect over the IP 1 interface - additional to SIGSTD)

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

Example 4 (Allow everything from IP address xx over IP 2 - additional to SIGSTD)

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

Example 5 (Self configuration without Sigmatek standard. Allow only the Lasal and a vnc client to connect)

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

Example 6 (Remove a INPUT rule)

Remove the following rule (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
```

Example 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"
```

If you search the internet for iptables or netfilter, you will find a lot of examples.

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



1.4.5.8 IP

Displays and sets the current TCP/IP settings.

Syntax

IP / IP INFO

Displays TCP/IP information; Ethernet Interface Number, Ethernet Chip, Addresses, Link status and if a server is running, connected clients are shown.

IP SHUTDOWN

Shuts down TCP-Debug-Server. Make sure all clients are offline. Note that this function blocks until the server is completely down. After the call, no TCP clients can connect until a restart is done, but you have the possibility to change the hosts IP address while the server isn't running. (See [SET IP](#) ...)

IP RESTART

Restarts the TCP debug server after a shutdown.

IP KILLCLIENTS

Kills all connected clients.

IP LOOPBACK

Installs and/or uninstalls an IP loopback interface.

IP DEBUG

Displays TCP server information on the screen.

IP LOG

Writes TCP server information to file.

IP TCPRETRANS <min> >max>

Shows and sets the lower and upper limit of the TCP Retransmission Timer in milliseconds for the specified Ethernet interface. If <min> and / or <max> are zero, the current settings are displayed. If <min> and / or <max> are not equal to zero, then the limits are set to the specified values. Default is 20 and 60000 ms. This command is only available for controls with the RTK operating system. This command can be executed at runtime or in the file Autoexec.lsl. But not via Remote CLI.

More than one Ethernet Interface

All commands can be used with the following additions. Without additional information, the first Ethernet interface is always used.

IP X (+ command)

X is the number of the Ethernet interface (1 .. first, 2 .. second, ...).

IP ALL (+command)

For all Ethernet interfaces.

Example

IP SHUTDOWN

Shuts down TCP server.

IP RESTART

Restarts TCP server.

IP INFO

Shows the TCP settings (if more Ethernet interfaces, the first).

IP ALL INFO

Shows the TCP settings of all Ethernet interfaces.

IP 2 SHUTDOWN

Shuts down TCP server 2.

IP ALL RESTART

Restarts all TCP servers.

IP 1 TCPRETRANS 0 0

Shows for the Ethernet interface 1 the lower and the upper limit of the TCP retransmission timer.

IP TCPRETRANS 0 900

Shows for Ethernet interface 1 the lower limit of the TCP retransmission timer and sets the upper limit to 900 ms.

1.4.5.9 IPCINFO

Displays information about the IPC.

Syntax

IPCINFO

1.4.5.10 SCREENSAVER

This command is not available for the RTK operating system. It displays or changes the screen saver settings.

Syntax

SCREENSAVER <x>

SCREENSAVER INFO

Options

<>	Time from the last touch of the display to activation of the screen saver in seconds. 0 restores the default setting of 3600 seconds.
-999	Deactivates the screen saver. Attention: Without a screen saver, the service life of the display is reduced and there is a risk that the display will be permanently damaged by images that have not been moved for a long time (so-called "burn-in").
INFO	Shows the set value.

1.4.5.11 SET

Displays and changes operating system options.

Syntax

SET

Displays current LASAL OS settings.

1.4.5.12 SET APPCODE

Displays or changes the size of the application code memory. The application data memory will be set to the value: total memory – application code memory.

RTK This command cannot be executed via remote CLI, but only as a CLI command (e.g. via USB keyboard or via VNC). To change the memory size, the command can also be executed in the autoexec.lsl file.

Salamander This command cannot be executed via remote CLI. To change the memory size, the command in the autoexec.lsl file must be executed. To display the memory size, a CLI command can be executed (e.g. via USB keyboard or via VNC). Also see [INFO](#).

Syntax

SET APPCODE size

size – the size of the application code memory in kilobytes, minimal 512

Example

SET APPCODE 4096

This command sets the application code memory to 4 MB.

1.4.5.13 SET APPDATA

Displays or changes the size of the application data memory. The application code memory will be set to this value: total memory – application data memory.

This command is only available with the RTK operating system. This command cannot be executed via remote CLI, but only as a CLI command (e.g. via USB keyboard or via VNC). To change the memory size, the command can also be executed in the autoexec.lsl file.

Syntax

```
SET APPDATA size
```

size – the size of the application data memory in kilobytes, minimal 512

Example

```
SET APPDATA 4096
```

This command sets the application data memory to 4 MB.

1.4.5.14 SET APPLSAVE

Configures the call of IF_PowerDown (integral component of lse_rtk) upon below mentioned events.

The RESET and the UPS event triggers the call of IF_PowerDown without further attention.



Syntax

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

Enables/disables the call of IF_PowerDown in case of a DIAS error, a watchdog error or a real-time runtime error.

1.4.5.15 SET BOOTTEXT

Changes the text when an application has been started.

If SET BOOTTEXT is not set, “Running Loader ...” is displayed.

Syntax

```
SET BOOTTEXT text
```

text - the text you want to display, maximal 47 characters

Example

```
SET BOOTTEXT
```

no boottext is displayed

```
SET BOOTTEXT now booting
```

boottext “now booting” is displayed

1.4.5.16 SET BTRUNTIME

Displays or sets background runtime counter.

Syntax

```
SET BTRUNTIME
```

Displays the background runtime counter.

```
SET BTRUNTIME value
```

Sets the background runtime counter.

value – number of ticks (1 tick = 10 ms), range: 0-4294967295, 0 = DISABLED

When the runtime counter is enabled, a watchdog checks the execution time of the background task (Background method). When the execution time of the Background task exceeds the limit specified by the runtime counter, the application stops with status BTRUNTIME.

Default: 0 = DISABLED

1.4.5.17 SET CAN

Displays or changes parameters of the CAN interface.

Syntax

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

ifnum - The number of the CAN-Interface (1 = CAN1, 2 = CAN2, ...).

baudind - index to the Baudrate-Table:

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 - The number of the CAN-Station (0-31).

```
SET CAN
```

Displays the settings of all CAN ports.

```
SET CAN ifnum BAUD baudind
```

Sets the CAN baudrate.

```
SET CAN ifnum STATION addr
```

Sets the CAN Station number.

```
SET CAN ifnum ONLCOM mode
```

Sets a flag to enable or disable the online communication via CAN.

mode - 0 - Disable the CAN online communication

1 - CAN online communication possible (default value)

Example

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

This command sets the baudrate of CAN interface CAN1 to 615 Kb and the station number to 23.

1.4.5.18 SET CLI

Sets the Command Line Interface (CLI) mode.

Syntax

SET CLI mode OFF|ON

Mode is either RUN, RESET or ERROR

RUN	Opens/closes the CLI on RUN (default OFF). The value ON is only useful, if the running project makes no output to the screen because otherwise the CLI output and the project output would interfere.
------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

RESET	Opens/closes the CLI on RESET (default ON).
--------------	---------------------------------------------

ERROR	Opens/closes the CLI on EXCEPTION (default ON).
--------------	-------------------------------------------------

Example

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

1.4.5.19 SET CPUBT

This command specifies on which CPU core the Background task should run.

Syntax

SET CPUBT <CPU number>

Numbering starts with 1.

1.4.5.20 SET CPUTCT

This command specifies on which CPU core the Cyclic task should run.

Syntax

SET CPUTCT <CPU number>

Numbering starts with 1.

1.4.5.21 SET CPURT

This command specifies on which CPU core the Realtime task should run.

Syntax

```
SET CPURT <CPU number>
```

Numbering starts with 1.

1.4.5.22 SET DATE

Changes date setting.

Syntax

```
SET DATE
```

This command has no parameter, you are prompted by the system to input valid values for year (range from 1990 to 2199), month, day and day of the week (0...Sunday, 1...Monday, ...).

1.4.5.23 SET DBGLEVEL

Displays or sets configuration of Debug messages.

Syntax

```
SET DBGLEVEL
```

Displays the configuration of Debug messages.

```
SET DBGLEVEL source level
```

Sets the configured Debug messages.

RTK Operating System

Source

Name	Description	Default level
CANLL	Low level CAN routines	0
APPHEAP	Application heap	0

GRAPHIC	Graphics library	0
SYSHANDLER	Exception handle	0
CAN	CAN communication	0
CDIAS	CDIAS library	0
TASKLISTS	Task list	0
BREAKPOINTINTS	Breakpoint handling	0
FLASH	CIPC flash module	0
IP	TCP/IP communications	0
TGRAPH	LARS graphics	0
SERIAL	Serial interface library	0
USB	USB interface library (default level = 2)	2
DEBUGIP	Loader: Debug Interpreter	0
MODLOAD	Module loader	0
KERNEL	LasalOS Kernel (Maintask, Serviceprovider)	0
LINKER	Linker library	0
LOADER	Loader library	0
TASKS	Task handling	1
VNCCLI	VNC client	1
VNCSVR	VNC server	1
LSLFILE	LslFile library	0
LSCMD	LslCmd library	0
WEBSVR	WEBSVR library	0
FILESYS	FILESYS library	0
DHCPCLI	DHCPCLI library	0
FTPSERVER	FTPSERVER library	1
LRM	LRM library (Lasal Remote Manager)	0

VARAN	VARAN library	1
VARCAN	VARCAN library (CAN devices)	0
VARSER	VARSER library (serial devices)	0
OSZI	OSZI library (Data Analyzer)	0
SAFETY	SAFETY library	0
SRAMDISK	SRAMDISK library	0
ETHERCAT	ETHERCAT library	0
HWTREE	Hardwaretree library	0
SDIAS	SDIAS library	0
LSLRPT	LASAL Repeater library	0
TRACE	Debugger trace	0
TCP	TCP/IP library	0

Level

The higher the level, the more detailed the messages.

- | | |
|---|--------------------------------------|
| 0 | no logging |
| 1 | error logging |
| 2 | additional debug information logging |
| 3 | extended debug information logging |

Salamander Operating System

Source

Name	Beschreibung	default level
TRACE	Debugger trace	0
ECATM	EtherCAT Master	0

ECATM_P_IMG		0
ECATM_E_VENT	EtherCAT Master events	0
ECATM_A_SYNC	EtherCAT Master asynchronous task	0
ECATM_MON	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
KEYEVENT_TS	Key events	1
XEVENTS	X-server events	1

HARDWARE-IO		0
IXAGENT	Software agent for IXON cloud. Siehe Fußnote 1	1
VPN	Open-VPN library	1
PDF_PRINTER	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

The higher the level, the more detailed the messages.

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

IXAGENT

The debug level must be set before the IXAGENT is started.

The Ixagent uses log levels of 0-9 (according to RFC3164) in contrast to the possible settings of 0-5. This results in the following allocation:

Debug level SIGMATEK	Debug level IXAGENT
5	9
4	7
3	6

2	4
1	3
0	2

1.4.5.24 SET DIASERROR

Displays or changes Dias error setup.

Syntax

SET DIASERROR

Displays current Dias error setting.

SET DIASERROR ENABLE | TRUE | ON

Enables DiasError interrupt.

SET DIASERROR DISABLE | FALSE | OFF

Disables DiasError interrupt.

Example

SET DIASERROR ON

1.4.5.25 SET DISPLAY

Shows or changes the color calibration. If the command is called without additional options, it shows the current settings after the call. If additional options are specified, the corresponding color parameter is set / changed.

This command is only available in the Salamander operating system!



Syntax

SET DISPLAY

Displays the current color calibration.

SET DISPLAY BRIGHTNESS xx

Sets the brightness of the display to xx. The value can be set between 0 (extremely dark) and 100 (extremely bright). If the value is outside this range, it is set to 100. The default value is 50.

SET DISPLAY BRIGHTNESS xx

Sets the color saturation of the display to xx. The value can be set between 0 (extremely pale) and 100 (extremely saturated). If the value is outside this range, it is set to 100. The default value is 50.

SET DISPLAY CONTRAST xx

Sets the color saturation of the display to xx. The value can be set between 0 (extremely low contrast) and 100 (extremely high contrast). If the value is outside this range, it is set to 100. The default value is 100.

SET DISPLAY HUE xx

Sets the hue / shade of the display to xx. The value can be set between 0 and 360. If the value is outside this range, it is set to 360. The default value is 0.

Example

SET DISPLAY BRIGHTNESS 75

The screen brightness is set to 75.

1.4.5.26 SET DISPLAYRESOLUTION

Allows changing the system resolution, respectively, switching from the text mode to the graphic mode.

Syntax

SET DISPLAYRESOLUTION x

x - 0 - Text Mode (no graphic display possible)
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) resolution available from LasalOS 01.02.150

(2) resolution available from LasalOS 01.03.010

(3) resolution available from LasalOS 01.02.145

Example

```
SET DISPLAYRESOLUTION 2
```

The CPU now runs in graphic mode with a resolution of 640 x 480 pixel.

Remarks

An OS Version number greater or equal to 01.02.130 is required. In activated graphic mode the memory requirement of the operating system (320 x 240 => + 150kB; 640 x 480 => + 600kB) increases. This should be considered when adjusting the OS HEAP.

1.4.5.27 SET DISPLAYROTATE XX

The display output can be rotated clockwise by 90°, 180° or 270° (VGA XGA). The command has to be set only once. After it was successfully executed, the system will reboot automatically.

Syntax

```
SET DISPLAYROTATE degrees  
degrees - 90° / 180° / 270°
```

Example

```
SET DISPLAYROTATE 90
```

The display output will be rotated clockwise 90°.

Requirements

An OS version number greater or equal to 01.02.022 is required.

1.4.5.28 SET FIRSTUSBDRIVE

Specifies the first drive letter to use for a USB drive.

The command must be inserted in the autoexec.lsl file.

This command is available for all controls with the Salamander operating system and for the RTK operating system when running on a control with X86-compatible EDGE technology. See hardware description of the control.

With the RTK operating system, the drive letter can be set in the range from E: to F:.

With the Salamander operating system, the drive letter can be set in the range from E: to Z:.

Syntax

SET FIRSTUSBDRIVE F:

Specifies the first drive letter for a connected USB stick with F:.

1.4.5.29 SET FORCEXTSINGLE

Permanently configures the Xtimer to the SINGLE_SHOT counter mode. This is useful for older applications that do not support the CONTINUOUS mode.

 If this flag is set in the Autoexec.lsl, the timer mode application cannot be changed later.

Syntax

SET FORCEXTSINGLE 1|0

Activates/deactivates the permanent SINGLE_SHOT mode of the Xtimer.

1.4.5.30 SET IP

Displays info or set IP-Parameters.

Syntax

SET IP

Displays information about the IP addresses.

SET IP HOSTADDR ip-address

Sets the Hosts IP-address.

SET IP SUBNET ip-address

Sets the Subnet mask.



The Subnet mask must be set before the Gateway address.

SET IP GATEWAY ip-address

Sets the Gateway IP-address.

SET IP DNS ip-address

SET IP DNS ip-address_2

For OS RTK: Sets the first and optional second DNS server address. The command without any IP will show up the settings.

SET IP DNS ip-address[ip-address_2]

For OS Salamander: Sets the first and optional second DNS server address. The command without any IP will show up the settings.

Requires (OS > 01.02.160)

SET IP PORT

Shows the set online port.

SET IP PORT port-address

Enables setting the online port (Default = 1954)

SET IP SPEED Mbps

Enables setting the data transfer rate (e.g 100 Mbit/s or 1000 Mbit/s). In the autoexec.lsl file, the speed must be specified after the command "SET IP X HOSTADDR ip-address".

SET IP X HOSTADDR ...

X: 1 ... first ethernet interface, 2 second ...

SET IP ONLTIMEOUT seconds

Sets the timeout for the TCP/IP online connection. When no activity is at the online connection for more than the specified timeout value, then the connection is closed.

SET IP IRQHOLDOFF x

x 1 ... 7 Cycles set to cyclic task priority

Ethernet interrupts can be locked over several periods to avoid runtime errors. The number of cyclic task cycles is transferred for which the IP Task / Interrupt is locked. The system load is thereby reduced because of incoming TCP/IP packets.



This has negative effects on the TCP/IP communication, since it is thereby slower (LasalOS > 01.01.248).

Example

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
```

Remarks

Before changing any of these values the TCP-Server must be shut down with command IP SHUTDOWN. After changing the server can be restarted with IP RESTART.

1.4.5.31 SET IP PORT

This method can be used to set or change the port for a specific Ethernet interface.



- Port 1000 is reserved for the Base server
- Port 1955 is reserved for the Comlink Command server
- Port 1956 is reserved for the Comlink Refresh server
- Port 1957 is reserved for the Alarm LSE server
- These ports cannot be used!

1.4.5.32 SET KEYS

Displays or changes current keyboard setting.



This setting can be changed at any time by pressing CTRL-ALT-F1 for US and CTRL-ALT-F2 for German keyboard layout.

Syntax

```
SET KEYS
```

Displays current keyboard setting.

```
SET KEYS US
```

Sets US keyboard layout.

```
SET KEYS GERMAN
```

Sets German keyboard layout (default).

```
SET KEYS NO_LANG_HOTKEYS
```

Do not intercept left Ctrl-Alt-F? keys to switch keyboard language.

Example

```
SET KEYS GERMAN
```

1.4.5.33 SET LASALCOM

Selects the COM-Port (RS232) that is used for online communication and sets the parameters of the selected interface.

Syntax

```
SET LASALCOM NONE
```

Disables online communication with RS232.

```
SET LASALCOM port BAUD baudrate
```

Re-enables online communication.

port - COM-Interface: COM1 | COM2

baudrate - 9600

14400

19200

38400

56000

115200

Example

```
SET LASALCOM COM1 BAUD 56000
```

Remarks

On Platform IPC and CIPC the LASALCOM Service is started on COM1 also when not set in autoexec.lsl! In order to use COM1 for a different device (e.g. for a touch panel) you can either redirect LASALCOM to a different COM (e.g. SET LASALCOM COM2) or disable the service (SET LASALCOM NONE). If you don't do this, the device on COM1 won't work properly.

1.4.5.34 SET LCD

Displays or changes the current LCD display settings.

This command is not supported on all platforms.



Syntax

SET LCD

Displays current LCD settings.

SET LCD BRIGHTNESS value

Sets the LCD's brightness (0-255).

SET LCD CONTRAST value

Sets the LCD's contrast (0-255).

Example

SET LCD BRIGHTNESS 128

SET LCD CONTRAST 128

1.4.5.35 SET MAINTIMER

Displays or sets the MainTimeBasis.

Syntax

SET MAINTIMER

Displays the timeslot for MainTimeBasis.

SET MAINTIMER value

Sets the timeslot for MainTimebasis.

value - valid durations in microseconds for the operating system RTK are: 50, 100, 125, 200, 250, 500, 1000 or 2000

Requirements

CIPCs requires Xilinx Version 2.1 or later

1.4.5.36 SET MOUSE

Displays or sets mouse/touch type and parameters.

Syntax

SET MOUSE

Displays current mouse settings

SET MOUSE [TYPE mousetype] [parameter]

Sets mouse type and parameters

Mouse Type

NONE	No mouse
PS/2	Mouse on PS/2 interface
COM1	Mouse on COM1 interface
COM2	Mouse on COM2 interface
COM3	Mouse on COM3 interface
USB	Mouse on USB interface
TOUCH TERMINAL	Touch (old hardware) on COM3 - see below
TOUCHPS/2	Touch on PS/2 interface
HAMPSHIRE	Touch (new hardware) on COM3 - see below
HAMCOM2	Touch (new hardware) on COM2 - see below
HAMUSB	Touch (new hardware) on USB - see below

Parameter

PORT adr	adr - Port base address (hexadecimal format)
IRQ irqnum	irqnum - Port interrupt number
BAUD baudrate	baudrate - Port baudrate (only for touch!)
CURSOR style	style - Cursor Style (0-6)



Please refer to your hardware documentation to find out whether you have an old hardware version (without Hampshire) or a new hardware version (with Hampshire).



In order to use the virtual touch keyboard (e.g. in an C-IPC's CLI) a valid touch interface has to be configured, as well as placing the *.sml Files into the c:\lslsys directory. These files can be found on the Master-CF-card.

Example

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

For USB devices and all touch-devices except HAMPSHIRE, the parameters PORT, IRQ and BAUD are ignored. HAMPSHIRE defaults to COM3, this setting can be overridden.

Requirements

LasalOS: all USB related stuff requires LasalOS 5.42 or later.

1.4.5.37 SET MULTICOREOBJS

Shows or sets number of additional processor cores running own realtime and cyclic tasks.

There are always one realtime and one cyclic task running on one, typically the first processor core of a CPU. On multi-core CPUs Salamander and Gecko OS support the execution of multiple realtime and cyclic tasks. When used in the file AUTOEXEC.LSL the command 'SET MULTICOREOBJS <num_cores>' sets the number of additional processor cores running their own realtime and cyclic tasks. With this command it cannot be determined which cores execute these tasks. The command can also be used via (remote) CLI to determine the current number of additional processor cores running their own realtime and cyclic tasks.

Syntax

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

Argument <num_cores> has to be integer ranging from zero to number of processor cores available to OS minus one; usage is allowed in AUTOEXEC.LSL only, where it is

mandatory; total numbers of realtime and cyclic tasks running on system are each given as <num_cores> + one.

Independently of number of processor cores available to OS there can be not more than seven additional cores running own realtime and cyclic tasks, totalling in maximum number of eight realtime and cyclic tasks runnable on CPU.

When used (without argument) on (remote) CLI command returns current setting.

1.4.5.38 SET MULTI_VM

This command is available only with the Salamander operating system. When used in the file AUTOEXEC.LSL SET MULTI_VM [ON|1|OFF|0] enables/disables the OS support for multiple VARAN managers (VMs) as far as multiple VMs are available in hardware. The command can also be used via (remote) CLI to determine the current state of support for multiple VMs and the number of VMs initialized by the OS.

Syntax

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

Argument is allowed in AUTOEXEC.LSL only, where it is mandatory.

When used (without argument) on (remote) CLI command returns current setting.

Requirements

Salamander OS V09.02.050 or later on x86 PLCs (usage on (remote) CLI supported starting with Salamander OS V09.07.112)

Salamander OS V09.03.120 or later on ARM PLCs (usage on (remote) CLI supported starting with Salamander OS V09.04.080)

1.4.5.39 SET OSHEAP

Displays or sets the size of OSHeap.

Syntax

```
SET OSHEAP
```

Displays the OSHeap information

```
SET OSHEAP heapsize
```

Sets the OSHeap properties

heapsize - specify length in kBytes

SET OSHEAP DEFAULT

Sets the original value

Requirements

LasalOS: Requires LasalOS 01.01.220 or later.

1.4.5.40 SET OSZIMEM

Displays or sets the size of memory for the DataAnalyzer.

RTK the default memory size is 512 kB. The memory can be set in a range from 512 kB to the size of APPDATA.

Salamander the default memory size is 100 kB. The memory can be set in a range from 100 kB to 4194303 kB.

Syntax

SET OSZIMEM

Displays the size of memory for the DataAnalyzer in kBytes.

SET OSZIMEM value

Sets the size of memory for the DataAnalyzer in kBytes.

Value - Length \geq 512 kBytes to APPDATA (operating system RTK)

Value - Length \geq 100 kBytes to 4194303 kBytes (operating system Salamander)

Remarks

The RTK operating system creates the memory in the application data memory area APPDATA.

Requirements

LasalOS: Requires LasalOS 01.02.45 or later.

1.4.5.41 SET PRINTER

Enables printer support and registers the printer interface.

Syntax

SET PRINTER

1.4.5.42 SET RAM

Displays or sets parameters for the RAM backup file processing (SRAM.DAT / ACTIVE.DAT).

Syntax

```
SET RAM
SET RAM DISPLAY
```

Displays the current settings for the RAM backup file processing.

```
SET RAM LOG ON|OFF
```

Switches logging of RAM backup file processing on or off. When logging is on, messages are written to C:\SRAM.LOG or C:\ACTIVE.LOG. The default value is OFF.

```
SET RAM RESET ON|OFF
```

If RAM RESET is ON, then the retentive data is backed up when the application is reset. The default value is ON.

```
SET RAM POWERDOWN ON|OFF
```

If RAM POWERDOWN is ON, then the retentive data is written when a power down is recognized. The default value is ON.

```
SET RAM FORMAT OLD|NEW
```

Sets the format of the RAM retentive data backup file. The default value is OLD. It is highly recommended that the value is set to NEW when the application was created with a Lasal version 0.60 or higher.

1.4.5.43 SET ROUTINGTABLE

Sets RoutingTable from a specified filename.

Syntax

```
SET ROUTINGTABLE
```

The RoutingTable will be cleared

```
SET ROUTINGTABLE value
```

Sets RoutingTable from a specified filename.

value - <filename.ext>

1.4.5.44 SET RTRUNTIME

Displays or sets real-time runtime counter.

Syntax

SET RTRUNTIME

Displays the real-time runtime counter.

SET RTRUNTIME value

Sets the real-time runtime counter.

value – number of maintimer ticks, range: 0-255, 0 = DISABLED

The time per maintimer tick can be set with the CLI command [SET MAINTIMER](#).

1.4.5.45 SET RUNTIME

Displays or sets the runtime counter.

When the runtime counter is enabled, a watchdog checks the execution time of a cyclic task (cywork method). When the execution time of a cyclic task exceeds the limit specified by the runtime counter, the application stops with status RUNTIME.

Syntax

SET RUNTIME

Displays the runtime counter.

SET RUNTIME value

Sets the runtime counter.

value – number of ticks (1 tick = 10 ms), range: 0-255, 0 = DISABLED

1.4.5.46 SET SECURE

Displays information or set options for the secure online channel.

Syntax

SET SECURE

Displays information about secure online channel.

SET SECURE VERIFYCLI on/off

Enables or disables the client certificate verification.

SET SECURE CERTPATH path

Sets the path for the certificate storage.

SET SECURE PORT port

Enables setting the secure online port (default = 2054).

1.4.5.47 SET SRAMRETAIN

Displays or sets the size of allocated SRAM memory for global variables.

Syntax

SET SRAMRETAIN

Displays the size of allocated SRAM memory for global variables.

SET SRAMRETAIN value

Sets SRAM memory for global variables.

value - specify length in bytes

Requirements

LasalOS: Requires LasalOS 01.01.161 or later

1.4.5.48 SET STATION

Displays or sets Station ID for a CPU.

The identifier must be set before setting the routing table!



Syntax

SET STATION

Displays the Station ID for a CPU.

SET STATION value

Sets the Station ID for a CPU.

value – range: 0-254

1.4.5.49 SET STOPWAIT_TIME

Sets the maximum time that the operating system waits after a [REBOOT](#) command from the application, until retentive server file and RamEx file data are written to file before the [REBOOT](#) command is run.

Syntax

```
SET STOPWAIT_TIME value
```

Sets the maximum wait time in milliseconds. A value of 0 means that the wait function is inactive

Requirements

The function for waiting until all retentive sever and RamEx file data are written, is available starting from the following versions:

LasalOS starting with 01.03.110

Loader starting with 02.02.186

Ramex starting with 1.1

1.4.5.50 SET TIME

Changes current time setting.

Syntax

```
SET TIME
```

This command has no parameter. You are prompted by the system to input valid values for hour, minute and second.

1.4.5.51 SET VISU

Sets the priority of the visualization task.

Syntax

```
SET VISU HIGH|LOW
```

The visualization runs in the background task. With this command you can change the priority of the background task to a higher priority. The default setting is LOW.

1.4.5.52 SET WATCHDOG

Configures the watchdog.

Syntax

SET WATCHDOG ON|OFF

Enables/disables watchdog.

SET WATCHDOG interval

Sets watchdog interval in ms , 0 < interval ≤ 255

1.4.5.53 TOUCHCFG

Loads the touch calibration values from a C:\TOUCHDAT.CFG file.



In order to use the virtual touch keyboard (e.g. in an C-IPC's CLI) a valid touch interface has to be configured, as well as placing the *.smi files into the c:\lslsys directory. These files can be found on the master CF card.

Syntax

TOUCHCFG

1.4.5.54 TOUCHTEST

Loads the touch calibration values from a C:\TOUCHDAT.CFG file and starts a simple touch test.

Syntax

TOUCHTEST

1.4.5.55 UPDATETOUCH

Updates a touch controller

- Netix to RTK: To update the configuration of multi-touch from Netix, 'NETIXCFG' must be entered as the parameter and the path to the configuration file specified.
The configuration file must be in OBP format (RAW).

- EGalaxy in Salamander: for the update, the directory path must be specified. The update tool (eUpdate2_ARM), firmware file (fw.bin) and the configuration file (SensorTestDefault.ini) must be stored in memory.

Syntax

```
UPDATETOUCH NETIXCFG c:\config.raw
UPDATETOUCH f:\egalaxy\
```

1.4.5.56 US

Sets date/time display to US format.

Syntax

US

To switch back to the European format, use the [EURO](#) command.

1.4.6 Batch Processing Commands

1.4.6.1 CLS

Clears the screen.

Syntax

```
CLS
```

1.4.6.2 DELAY

Delays the specified time.

Syntax

```
DELAY time
```

Delay time in seconds.

1.4.6.3 ECHO / @ECHO

Outputs the text following the keyword to the console.

@ECHO suppresses the output of the source-line of the batch file and outputs only the text.

Syntax

```
ECHO ... text ...
@ECHO ... text ...
```

1.4.6.4 EXIT

Stops execution of the batch file.

Syntax

```
EXIT
```

1.4.6.5 GOTO

Continues execution of the batch file at Label "label", which must be preceded by a ":".

Syntax

```
GOTO label
```

Example

```
@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

Conditional execution of a CLI-command, depending on the existence of a file.

Syntax

IF [NOT] EXISTS filename cli-command
filename - Must be in 8:3 format.

cli-command - Any valid CLI-command.

1.4.6.7 PAUSE

Suspends processing of a batch program and displays a message that prompts the user to press any key to continue.

Syntax

PAUSE

1.4.6.8 REM

Enables you to include comments in a batch file or in your AUTOEXEC.LSL file. The REM command is also useful for disabling commands.

Syntax

REM [string]

Example

```
rem SET MOUSE TYPE TOUCH
Old cmd line.

rem Now I don't use the touch anymore
Comment.

SET MOUSE TYPE NONE
New cmd line.
```

1.4.7 Enviroment Variables

1.4.7.1 FILENAMES

Determines how file names are shown with CLI command DIR.

Syntax

SETENV FILENAMES SHORT

This is the default setting. Files are shown with their short 8.3 name.

SETENV FILENAMES LONG

Files are shown with their full name (up to 256 characters).

1.4.7.2 REBOOTONPF

Determines if the PLC is rebooted after a power failure.

Each power failure is written to the system log.

Syntax

SETENV REBOOTONPF 1

This is the default setting. At power-failure the application is stopped immediately, then the PLC waits until power-failure has gone and then reboots.

SETENV REBOOTONPF 0

No special power-fail action.

1.4.8 LasalOS File Error Return Codes

Number	Error code	Description
0	NO_ERROR	Not an error. This value indicates success of an operation.
-1	ERROR_RESERVED	This error code is reserved and will not be returned by the FileI/O-Class.
-2	PARAM_ERROR	The parameters passed to an FileI/O-Class function are invalid. For example, the flags passed to RTFOpen are contradictory or the size of a string buffer is too small.
-3	INVALID_FILENAME	A device, directory, or file name passed to the FileI/O-Class has an invalid syntax, contains illegal characters, or exceeds MAX_PATH (80) characters.
-4	DRIVE_NOT_FOUND	The program attempted to access a logical drive which is not mounted.
-5	TOO_MANY_FILES	The program attempted to open more files than slots were available in the file table.

-6	NO_MORE_FILES	This value is returned by RTFFindFirst and RTFFindNext when no more files satisfy the search criteria.
-7	WRONG_MEDIA	A diskette has been replaced in a diskette drive, and the serial number of the new disk does not match the serial number of the original disk. To correct this error, the original diskette must be inserted or the operation must be failed.
-8	INVALID_FILE_SYSTEM	The information found in the boot record of a logical drive is inconsistent and probably corrupted. The drive cannot be mounted.
-9	FILE_NOT_FOUND	The requested file name was not found on the disk.
-10	INVALID_FILE_HANDLE	A file handle passed to an FileI/O-Class API function is invalid. Either it has been closed already or it was not returned by a previous successful call to RTFOpen or RTFFindFirst, or it was closed automatically due to the removal of the media hosting the file.
-11	UNSUPPORTED_DEVICE	A device in the device list specified a device other than DEVICE_DISKETTE or DEVICE_FDISK in the DeviceType field.
-12	UNSUPPORTED_DRIVER_FUNCTION	This error is returned by device drivers or system drivers. For example, a device driver for read only devices would return this error on attempts to write to the device.
-13	CORRUPTED_PARTITION_TABLE	The FileI/O-Class has found inconsistent values in a device's partition table. The device cannot be mounted.
-14	TOO_MANY_DRIVES	The number of logical drives found on all devices given in the device list exceeds the FileI/O-Class's internal drive table.
-15	INVALID_FILE_POS	A call to RTFSeek attempted to position the file pointer before the start of the file.
-16	ACCESS_DENIED	This error is returned whenever a security check fails. A few such checks are: <ul style="list-style-type: none"> Attempt to open a file for read/write access, but the read-only attribute is set. Attempt to open an already open file, and at least one of the file instances requires write access, and SHARED is not specified for all instances of the file. Attempt to create a file and the specified file already exists with the read-only attribute set. Attempt to open a volume label. Attempt to open a directory with read/write access. Attempt to open a directory with flag OPEN_NO_DIR. Attempt to delete a file with attribute read-only, volume label, or directory set. Attempt to rename a file across drives. Attempt to rename a volume label. Attempt to rename a directory to one of its child directories.

		<ul style="list-style-type: none"> Attempt to rename the current directory. Attempt to set attributes, date and time, or file size of a root directory. Attempt to change a volume label or directory attributes. Attempt to write to, truncate, or extend a file open for read-only access. Attempt to remove a directory which is not empty. Attempt to remove a directory with the read-only attribute set. Attempt to remove the root or the current directory. RTFResetDisk was called while files are open on the target drive.
-17	STRING_BUFFER_TOO_SMALL	The size of a string buffer passed to an the FileI/O-Class function is too small to hold the result.
-18	GENERAL_FAILURE	A device driver reported an error without supplying additional information about the cause. For example, non-existing hardware or fatal hardware failures could generate this error.
-19	PATH_NOT_FOUND	The path for a file or a directory passed to the FileI/O-Class was not found.
-20	FAT_ALLOC_ERROR	The FileI/O-Class has found invalid values in a partition's FAT. The FAT is most likely corrupted and the partition must be reformatted.
-21	ROOT_DIR_FULL	It was attempted to create a new file in a root directory, but the directory is full. Unlike subdirectories, root directories have a fixed size and cannot be extended.
-22	DISK_FULL	It was attempted to extend a file or directory, but not enough free clusters to satisfy the request were found. For function RTFExtend, this error is returned when not enough continuous clusters are found.
-23	TIMEOUT	This device error is reported when a device fails to respond within a reasonable period of time.
-24	BAD_SECTOR	A device driver has reported that a sector on the disk could not be read or written.
-25	DATA_ERROR	A device driver has reported that a sector read from disk has failed a data integrity check. Typically, data is stored with CRC check sums which are used for such checks.
-26	MEDIA_CHANGED	During a device access, the driver has detected that the media in the drive has been removed or exchanged.
-27	SECTOR_NOT_FOUND	A device driver was not able to locate a requested sector. Either a corrupted boot sector or corrupted low-level formatting can cause this error.
-28	ADDRESS_MARK_NOT_FOUND	The address mark normally written during a low-level format was not found during a disk read or write operation.
-29	DRIVE_NOT_READY	A disk device does not respond, either because it is not present, the media is not inserted, or because of a hardware failure.

-30	WRITE_PROTECTION	It was attempted to write to a write-protected media.
-31	DMA_OVERRUN	A DMA controller has reported this error. This can happen when a DMA buffer spans a 64k address boundary.
-32	CRC_ERROR	A CRC check failed during a device read or write operation.
-33	DEVICE_RESOURCE_ERROR	A device driver has reported that some resource it requires is not available. For example, the floppy driver was unable to allocate a DMA buffer or the RAM disk driver was unable to allocate space for new sectors.
-34	INVALID_SECTOR_SIZE	A device reported itself to be formatted with a non-standard sector size. Usually, FAT volumes should use sectors of 512 bytes size. Please contact On Time for information about how the FileI/O-Class can support other sector sizes.
-35	OUT_OF_BUFFERS	The FileI/O-Class was unable to allocate a new buffer in its internal buffer cache. This situation can only occur if all buffers are dirty and none of the dirty buffers reside on the same physical device for which a new buffer is required. To avoid this error, increase the number of buffers, close some files, or flush buffers before the failing function is called.
-36	FILE_EXISTS	This error is reported on an attempt to rename a file to an existing file name or create a directory with the name of an existing directory or file.
-37	LONG_FILE_POS	This return code does not constitute an error. It is returned by RTFSeek when the new file pointer value exceeds 231-1 (2G minus one). However, the function has succeeded when this value is returned. Use function RTFGetFileInfo to retrieve the actual file pointer.
-38	FILE_TOO_LARGE	The application has attempted to extend a file to contain 4G or more bytes. However, FAT file systems only support file sizes up to FFFFFFFFh bytes. This restriction also applies to FAT-32 partitions.

1.5 Software Configuration

1.5.1 Software Components

The LASAL operating system consist of the following distinct software components.

Operating system	The operating system consists of an image file and has the file extension .RTB (IPC, C-IPC) or .BIN (386er CPUs).
System data	System data are, for example, DLLs (Dynamic Loadable Library) or control data for the virtual keyboard of the IPC with touch located in the C:\LSLSYS directory of the target system.
Batch data and macros	Are located on the target system in the directory C:\LSLCMD.

LASAL Class project	The LASAL Class project files are located in the target system in the directory C:\LSLWORK. In LASAL Class, the files in this directory can be created with the 'Project / Create Bootdisk' menu command.
LASAL Screen (LSE) project	The LASAL Screen project files are located on the target system in the directory C:\MPC. The C:\IPC.INI also belongs to the LSE project. In the PC, the files are located in the project path under the RUNTIME subdirectory.
Firmware	The firmware for diverse components (e.g. Xilinx) of the target system can be updated with a software command.

1.5.2 Installation with the Debug Tools LASAL Class and LSE Operating System

The operating system can be updated in LASAL Class via the menu item, 'Debug / Extras / Download operating system'. Do not turn off the target system during operating system update!

System Files, Batch Files and Macros

In LASAL Class, these files can be transferred to the corresponding directory via the menu item 'Debug / Extras / Transfer file to SPS'.

LASAL Class Project

In LASAL Class, a project can be transferred into the program memory via the menu item 'Debug / Extras / Transfer file to SPS'. To save the project permanently, the menu item 'Debug / Extras / Save project in the control' must be selected.

LASAL Screen Project

In LASAL Screen, the LSE project is installed using the menu item 'Debug / Download project'.

Firmware

The debug tools cannot be used to update the firmware.

1.5.3 Installation via the Command Line Interface (CLI)

Operating System

The operating system can be installed with the CLI command, `BOOT <OS image file> [>DriveLetter]`. C : is the default value for <DriveLetter>.

System Files, Batch Files and Macros

Macros, system files and Batch files are installed by copying the desired files to the corresponding directory.

LASAL Class Project

A LASAL Class project can be installed by copying the files to the C:\LSLWORK directory.

LASAL Screen Project

An LSE project can be installed by copying the files to the C:\LSLWORK directory and by the C:\IPC.INI file

Firmware

A separate CLI command is available for updating the firmware.

An overview of the available CLI commands can be found in under [Command Line Interface](#).

1.5.4 Automatic Installation of a Exchangeable Storage Media

Exchangeable storage media (disk or USB drive) can be installed in two ways:

1. **In the selected media, an AUTOEXEC.LSL with an installation instruction stored that contains the drive letter for the search path of the AUTOEXEC.LSL.**

The search order for the AUTOEXEC.LSL on the IPC is as follows:

A:\(Floppy),

D:\(Smartmedia if available),

C:\(Hard disk).

The search order for the AUTOEXEC.LSL on the C-IPC is:

B:\(USB-Floppy),

E:\(first USB drive with a LUN),
D:\(Compact Flash in the 2nd slot),
C:\(Compact Flash in the 1st slot).

2. The AUTOEXEC.LSL on the C: drive contains commands that are used to start a batch file or a program from an exchangeable storage device. The path for Rexx programs is first extended with the specified drive letters. An attempt is then made to start a Rexx program. Finally, the search path is reset to the standard value.

The AUTOEXEC.LSL entry appears as follows:

```
SETENV REXX_MACROS B:\;F:\;G:\;H:\;I:\;J:\;K:\;L:\;M:\;N:\;O:\;  
REXX AUTOSTRT  
SETENV REXX_MACROS C:\LSCMD
```

Because the disk change can no longer be recognized, the Rexx program cannot be run from an installation disk. The following entry is therefore required when using a disk:

```
DEL c:\tmpcmd  
COPY a:\autostrt c:\tmpcmd  
REXX c:\tmpcmd  
DEL c:\tmpcmd
```

1.5.5 Rexx Programs for Installing Software

Rexx programs are available for the administration of software packages; a software package consists the following:

- The files to be installed and the corresponding description file.
- Different software components can be included, such as an operating system and LASAL Class project.
- The ability to access several types of storage media (if the disk is full, for example).
- An identification feature to recognize storage media that may have been inserted wrong.

Rexx Requirements

- LasalOS 5.42 or higher
- The rexx.dlm file must be located in the C:\LSLSYS directory.
- For compressed software packages, the lslzip.dlm file must be located in the directory C:\LSLSYS.

A detailed description of the Rexx interpreter can be found in the chapter [Rexx Interpreter](#).

1.5.6 Description File of a Software Package

The description file is an ASCII file named MEDIUM.INF, which consists of a header that describes the software package and additional entries for describing the individual files located on the storage medium.

The header is configured as follows:

```
NAME <Name of the software package>
DATE <Date>
DISKNBR <Number of the data media, starting with 1>
[ INCOMPLETE ]
```

The INCOMPLETE entry is optional and means that further data media belong to the software package.

Entries used to describe the files stored on the data medium consist of following:

```
<Type> <Path> <File> <Command> <Target path> <Group>
```

<Type>	Marks the file type.				
	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">FILE</td><td>This means the file is unpacked.</td></tr> <tr> <td>SPLITA RCH</td><td>This entry describes a file or a partial file of an archive. An archive is a file in which the source files are combined in packed form. PKG is the basis name (without file extension). The file extension is defined by the archiving program to mark the order of part files for example. The root directory on the data medium is the path to the archive file.</td></tr> </table>	FILE	This means the file is unpacked.	SPLITA RCH	This entry describes a file or a partial file of an archive. An archive is a file in which the source files are combined in packed form. PKG is the basis name (without file extension). The file extension is defined by the archiving program to mark the order of part files for example. The root directory on the data medium is the path to the archive file.
FILE	This means the file is unpacked.				
SPLITA RCH	This entry describes a file or a partial file of an archive. An archive is a file in which the source files are combined in packed form. PKG is the basis name (without file extension). The file extension is defined by the archiving program to mark the order of part files for example. The root directory on the data medium is the path to the archive file.				
<Path>	Path to the file on the data medium				
<File>	Name of the file on the data medium (without path) The following errors are only interpreted at the FILE type.				
<Command>	Defines what should happen to the file on the target system. <ul style="list-style-type: none"> • COPY • PRJINST • OSINST 				
<Target path>	A symbolic name for the directory on the target system. The effective value of this variable is determined not until time of installation on the target system. The following values are possible: <ol style="list-style-type: none"> 1. <Command>=COPY MPC_DIR IPCINI_DIR CMD_DIR LSLSYS_DIR 2. <Command>=PRJINST LSLWORK_DIR 				

	3. At the remaining values for <Command>, <Target path> is not interpreted.
<Group>	With this entry, files can be assigned to a group name. Thereby making it possible for a software package to contain files for several target systems. During installation, it can be specified that only files of a special group should be installed.

1.5.6.1 Examples for Description Files

1) A Lasal Class and LSE project on the IPC and a LASAL Class project in a DCL641:

The software components of the IPC are marked with the group name GRP_IPC and the ones of the DCL641 with the group name GRP_386.

```
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) A software package in archive format separated on two disks.

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 Program to Create an Archive File

With the ARCH.EXE windows program, the content of a directory can be copied to an archive file or in several partial files of an archive. When too little space is available on the storage medium, an archive must be separated in partial files.

Syntax

```
arch <archive name> [parameters]
```

-r<root-dir>	Root directory of <file-spec>, Standard=.
	The directory which contains the files that should be added to an archive (including all subdirectories)
-f<file-spec>	File name, can also contain wild cards (* and ?), Std=.*
-x	Extraction of the files from the archive to the root directory. Without this parameter an archive is created.
-s<splitsize>	The archive is separated in part files of the <splitsize> size.
-?	Display of the help

Example

```
ARCH example.arc
```

Creates an archive in the current directory with all files (*.*) from the current directory

```
ARCH C:\example.arc -rE:\Temp -f*.txt
```

Creates an archive in C:\ from all .TXT-files in E:\Temp

```
ARCH example.arc -x
```

Searches the current directory for the archive file example.arc and extracts the files to the current directory

```
ARCH C:\example.arc -rE:\Temp -x
```

Extracts the files from the archive (if available) to E:\Temp

1.5.8 AUTOSTRT REXX Program for installation software package

The AUTOSTRT Rexx program allows a software package to be installed. The installation parameters such as the address of the target system and target directories are also defined in the first part of the program, therefore it is necessary to edit the AUTOSTRT file and adapt the parameters. The second part contains the program code. This part must only be changed if a different program configuration is desired.

After the software package is created, the AUTOSTRT file is either copied to the first data medium of the software package or to a separate data medium when too little space is available.

If the data medium with the AUTOSTRT file is in the drive during boot-up and the AUTOEXEC.LSL contains an AUTOSTRT call, the installation starts automatically.

1.5.8.1 Example

1.5.8.1.1 System Update IPC

On the exchangeable storage medium, there is an AUTOEXEC.LSL and the operating system map (.RTB file).

The AUTOEXEC.LSL entry appears as follows:

```
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:\ls1sys\lasalos.rtb
```

If in the BIOS boot order, the A: drive is defined before the C: drive (BIOS entry, Boot Sequence A, C), an error message appears that says the wrong data medium is inserted because there is no bootable operating system on the installation disk. To avoid this error message, change the BIOS settings or insert the disk only after the operating system starts from C: (this means after the display 'loading program ...').

Waiting too long is not recommended, as the AUTOEXEC.LSL is executed from the C: drive.

After updating the operating system, it is necessary to reboot!

1.5.8.1.2 System Update CPU

A software package is created and on the installation disk, the MEDIUM.INF file, AUTOSTRT Rexx program and the operating system map (.BIN file) are located. The CAN station number of the disconnected CPU is 2.

The MEDIUM.INF file appears as follows:

```
NAME          OS-Update-DCL641-5.42
DATE          04/01/14,13:57:06
DISKNBR      1
```

```
FILE . DCL641.BIN PRJINST NA GRP_DCL
```

The 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.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      = 'CAN1'
settings.0.address        = '2'
settings.0.group          = 'GRP_DCL'

```

1.5.8.1.3 LASAL on Several Disks

- 1) All files belonging to the installation are to be copied in a separate directory, e.g. C:\INST\GRP_IPC.
 - a. Lasal Class files: In Lasal Class the files are created with 'Project / Create bootdisk' in the directory C:\INST\GRP_IPC\LSLWORK.
 - b. Lasal Screen files: The directory <LSE-Project>\RUNTIME is copied to C:\INST\GRP_IPC\RUNTIME.
 - c. The <LSE-Projekt>\RUNTIME\IPC.INI file is copied to C:\INST\GRP_IPC\IPCINI\IPC.INI.
- 2) Creation of the MEDIUM.INF description file. Content of the 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) Since the software package should be copied to disks, converting to an archive format is more practical. When converting to an archive format, all files belonging to the software package are put into an archive file, which is then packed and divided into smaller files that are optimally sized for a disk.

With the ARCH.EXE program, a complete directory can be packed and the separated archive files copied into a separate directory (C:\PKG1AR).

Program call:

```
MD C:\PKGAR  
ARCH C:\PKGAR\PKG.AR -xC:\INST -f*.* -s1400000
```

In the directory C:\PKGAR the files PKG.000, PKG001 and PKG.002 are located.

- 4) The files PKG.000, PKG.001 and PKG.002 are copied each to its own disk. For every disk a separate MEDIUM.INF file has to be created:

Disk 1:
PKG.000
MEDIUM.INF

Content of 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

Content of 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) Adjust parameter in the AUTOSTRT file. This file is copied then to the 1st installation disk.

Content of 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       = 'LOCAL'
settings.0.address          = 'NA'
settings.0.group            = 'GRP_IPC'
```

1.6 REXX Interpreter

1.6.1 What is LasalOS Rexx?

LasalOS Rexx is an application of the Rexx (Restructured Extended Executor) language, which is an interpreted script language that allows programs to be written in a structured form. Its purpose is to provide a way to write procedures for administrative tasks when no Lasal project is running, such as procedures for distributing software and data files from one PLC to another. However, it is not intended to be called from a Lasal project.

LasalOS Rexx is able to pass arbitrary command strings for execution in the CLI (Command Line Interface) environment. Several integrated LasalOS-specific functions are available in addition to the standard integrated Rexx functions.

1.6.2 The Purpose of this Document

This document describes the LasalOS Rexx language application and the integrated LasalOS-specific functions but is not a reference for Rexx. Please refer to other documentation for a description of the Rexx language.

Requirements

LasalOS	Requires LasalOS V5.42 or later
Platform	LasalOS Rexx is available on an IPC and in LARS

Limitations

Only filenames in the 8.3 format are supported.

Installation

On an IPC copy the file Rexx.DLM to the target system into the directory C:\LSLSYS. The file Rexx.DLM can be found in the Lasal program directory (usually C:\Program Files\Lasal) in the subdirectory LasalOS\Rexx. In LARS the Lars setup program automatically installs LasalOS Rexx.

Using LasalOS Rexx

- Choose a directory where you want to store your Rexx programs, e.g. C:\USRCMD.
- Edit AUTOEXEC.LSL and insert the line 'SETENV Rexx_MACROS C:\USRCMD;C:\LSLCMD'.
- Copy your Rexx program into C:\USRCMD.

- Execute the Rexx program (CLI command: Rexx <name of the Rexx program>).

Executing Rexx Programs

Rexx programs are executed by entering the following CLI command:

```
Rexx [<program>]
```

Where program is the name of the Rexx program to be executed. If no program name is specified, LasalOS-Rexx waits for Rexx commands to be typed in and will execute those commands when the end-of-file character (Ctrl-Z) is typed.

External Rexx Programs

LasalOS-Rexx searches for Rexx programs, using a combination of the Rexx_MACROS environment variable and the filename extensions. This rule applies to both external function calls and the program specified in the command line. When the user has to call a function, it is coded as follows.

```
Call myextfunc arg1, arg2
```

LasalOS-Rexx first looks for a file called myextfunc in the current directory. If it can't find that file, it looks in each directory specified in the Rexx_MACROS environment variable for a file called myextfunc. If the file is not found, LasalOS-Rexx then attempts to find a file called myextfunc.rex in the current directory. Each directory in Rexx_MACROS is then searched. LasalOS-Rexx continues next by appending .cmd to the supplied external function name followed by .rx. Only if a file does not exist in any directory in Rexx_MACROS either with the supplied filename or with a filename appended with .rex, .rx or .cmd, LasalOS-Rexx returns a message stating that the external function is unknown. To set environment variable, use the CLI command [SETENV](#).

```
SETENV <name of variable> [<value of variable>]
```

Example

```
/* 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 [Appendix A](#) you can find other sample REXX programs.

1.6.3 LasalOS-specific Built-in Functions

LasalOS REXX has additional functions used for communicating with a remote PLC (online functions) and other useful functions that are not part of the REXX language.

Considerations when Using Online Functions

Every online function has at least 2 parameters: interface and address. A remote PLC can be addressed by defining the interface type (CAN, IP, ..) and the address. The CAN interface address is the station number (0..31); the IP interface address is the IP number expressed in the Internet standard “.” (dotted) notation.

For every online function, an online connection to the remote PLC has to be established by calling the function [ONLINE\(\)](#). When the online connection is no longer needed,

[OFFLINE\(\)](#) should be called to free the resources. [ONLINE\(\)](#) returns an online descriptor, which is used as a parameter for the [ONLINE\(\)](#) function.

The connection can also be established by sending the interface-type and address directly to the online function instead of using the online descriptor, in which case, the online connection enabled and disabled implicitly in the online function. The advantage of this method is that the caller does not have to call the [ONLINE\(\)](#) or [OFFLINE\(\)](#) functions. The disadvantage however, is that a new online connection has to be established for each new online command and a great amount of resources is required. This method should be used if an online command is called at regular time intervals (when the CPU status is queried, for example).

The interface and address parameters can contain the following values:

Interface	Address
CAN1 or CAN	<CAN station number> (0-31)
CAN2 or CAN	<CAN station number> (0-31)
IP1 or IP	<IP-number> (dotted decimal format)
DESCR	<Descriptor-number returned from ONLINE>



Some commands accept a LOCAL interface type, in which case, the destination is the local PLC and not the remote PLC. The address parameter for a LOCAL interface is an arbitrary string (not an empty string).

Example

```
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 is used to list the contents of a directory.

Transfer parameters	Type	Description
directory		The name of the directory
stem		The output is written to the compound variable stem

Return parameters	Type	Description
		0 Function successful
		≠0 Error code

Variable stem

<stem>.0	Number of directory entries
<stem>.1-<stem>.n	Directory entries
<stem>.NAME	Name of matched file/directory, without the path
<stem>.SIZE	Length of file in bytes
<stem>.TIME_WRITE	Time of last write to file
<stem>.TIME_CREATE	Time of file creation (-1L for FAT file systems)
<stem>.TIME_ACCESS	Time of last file access (-1 for FAT file systems)
<stem>.A_ARCHIVE	Archive; set whenever the file is changed and cleared by the BACKUP command
<stem>.A_HIDDEN	Hidden file; not normally seen with the DIR command, unless the /AH option is used; returns information about normal files as well as files with this attribute
<stem>.A_NORMAL	Normal; file can be read or written to without restriction
<stem>.A_RDONLY	Read-only; file cannot be opened for writing, and a file with the same name cannot be created
<stem>.A_SUBDIR	Subdirectory
<stem>.A_SYSTEM	System file; not normally seen with the DIR command, unless the /A or /A:S option is used

1.6.3.2 DRIVEINFO

DriveInfo is used to list drive information on a remote station.

Transfer parameters	Type	Description
interface		
address		Address of the PLC (see Using Online Functions)
drive		Name of the drive

stem		Provides the name of the variable in which the information is written
Return parameters	Type	Description
		0 Function successful
		≠0 Error code

The stem variable contains the following parameters

Variable	Description
stem.BYTES_PER_SECTOR	Number of bytes per sector
stem.SECTORS_PER_CLUSTER	Number of sectors per cluster
stem.TOTAL_CLUSTERS	Number of total clusters
stem.FREE_CLUSTERS	Number of free clusters
stem.TOTAL_BYTES	Drive size in bytes
stemUSED_BYTES	Size of used memory in bytes
stemFREE_BYTES	Size of available memory in bytes

Requirements

This command is supported by Rexx.dlm starting with version 01.01.024. Under Salamander it is supported starting with OS version 09.03.080 and 09.02.020 respectively.

1.6.3.3 FILEINFO

FILEINFO is used to list information about a file on a remote station. The output is written to the compound variable <stem>.

Transfer parameters	Type	Description
Interface		
address		Address of the PLC (see using online functions)
filename		Name of the file
stem		The output is written to the compound variable stem
Return parameters	Type	Description
		0 Function successful

		#0 Error code
--	--	-------------------------------

Variable stem

<stem>.NAME	Name of matched file/directory, without the path
<stem>.SIZE	Length of file in bytes
<stem>.TIME_WRITE	Time of last write to file
<stem>.TIME_CREATE	Time of file creation (-1L for FAT file systems)
<stem>.TIME_ACCESS	Time of last file access (-1 for FAT file systems)
<stem>.A_ARCHIVE	Archive; set whenever the file is changed and cleared by the BACKUP command
<stem>.A_HIDDEN	Hidden file; not normally seen with the DIR command, unless the /AH option is used; returns information about normal files as well as files with this attribute
<stem>.A_NORMAL	Normal; file can be read or written to without restriction
<stem>.A_RDONLY	Read-only; file cannot be opened for writing, and a file with the same name cannot be created
<stem>.A_SUBDIR	Subdirectory
<stem>.A_SYSTEM	System file; not normally seen with the DIR command, unless the /A or /A:S option is used

1.6.3.4 GETCH

Gets a character from the console without echo.

Remarks

The GETCH() function reads a single character from the console without echoing. When reading a function key or an arrow key, GETCH() must be called twice. The first call returns 0 and the second call returns the actual key code.

Return Value

This function returns the ASCII-Code of the character.

1.6.3.5 GETCHE

Gets a character from the console with echo.

Remarks

The GETCHE() function reads a single character from the console and echoes the character read. When reading a function key or an arrow key, GETCHE() must be called twice. The first call returns 0 and the second call returns the actual key code.

Return Value

This function returns the ASCII-Code of the character.

1.6.3.6 GETCPUSTATUS

Retrieves the CPU status of a remote PLC.

Transfer parameters	Type	Description	
interface			
address		Address of the PLC (see Using Online Functions)	
Return parameters	Type	Description	
		≥ 0 CPU status code < 0 Error code	

1.6.3.7 GETDATAD

Gets data from the application data area of a local or remote PLC.

Transfer parameters	Type	Description	
interface		The interface type LOCAL is supported	
address		Address of the PLC (see Using Online Functions)	
mem-addr		Memory address relative to start of the application data area	
length		Length of the requested memory; only values from 1 to 256 are allowed	

stem		The output is written to the compound variable stem. <stem>.len contains the length, <stem>.0 the first byte and so on
Return parameters	Type	Description
		0 Function successful ≠0 Error code

1.6.3.8 GETOSVERSION

Gets the version number of the operating system. The usage of this function is deprecated, use [GETOSVERSION2\(\)](#) instead.

Transfer parameters	Type	Description
interface		The interface type LOCAL is supported
address		Address of the PLC (see using online functions)
Return parameters	Type	Description
		If no error occurs, the function returns the version of the operating system. Otherwise a negative error code is returned. The version number is returned in a decimal format, e.g. OS Version 5.42 is returned as the value 542. The formula for the returned value is as follows: ver_major * 100 + ver_minor



Since the new 3-digit format of the OS version number, this function does not return unambiguous values, e.g.:

- | | |
|--------------------|------------------------|
| OS version 5.44 | -> Return value = 544 |
| OS version 1.1.44 | -> Return value = 1744 |
| OS version 1.1.101 | -> Return value = 1801 |
| OS version 1.2.001 | -> Return value = 1801 |

1.6.3.9 GETOSVERSION2

Gets the version number of the operating system.

Transfer parameters	Type	Description
interface		The interface type LOCAL is supported

address		Address of the PLC (see Using Online Functions)
Return parameters	Type	Description

If no error occurs, the function returns the version of the operating system. Otherwise a negative error code is returned. The version number is returned in a decimal format, e.g. OS Version 5.42 is returned as the value 542.

The formula for the returned value is as follows: $(ver_major >> 4) * 100000 + (ver_major \& 0xf) * 1000 + ver_minor$

Example

- OS version 5.44 -> Return value = 544
 OS version 1.1.44 -> Return value = 101044
 OS version 1.1.101 -> Return value = 101101
 OS version 1.2.001 -> Return value = 102001

1.6.3.10 GETRES

Gets the graphic resolution of the CPU.

Transfer parameters	Type	Description
none		
Return parameters	Type	Description

The returned text is composed of two distinct values separated by a Comma ','.

1. the resolution
2. the rotation

Where resolution can be one of

VESA mode:

GNO: No Display
 G0: Textmode
 G10:Graphicmode 320x200x16
 G11:Graphicmode 320x200x256
 G12:Graphicmode 320x200x16M
 G20:Graphicmode 640x480x16
 G21:Graphicmode 640x480x256
 G22:Graphicmode 640x480x16M

		<p>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 Sigmatek mode: 128x64: 128x64x1 and rotation is defined by either "norot" (for no rotation) or "<code><direction + angle></code>" (for arbitrary rotation). Direction: CW or CCW Angle: e.g.: 90 So, CCW90 means a counter-clock wise rotation by 90°.</p>
--	--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1.6.3.11 HWINFO

Provides information on the control.

Transfer parameters	Type	Description
Info_dst		Provides the name of the variable in which the information is written
Return parameters	Type	Description
		0 Function successful ≠0 Error code

Otherwise, an error code is returned. The info_dst variable contains the following parameters:

Variable	Description
Info_dst.PLATFORM	The name of the platform, e.g. "C-IPC AMD LX800"
Info_dst.FPGAVER	The FPGA version
Info_dst.SERIAL	Serial number of the module
Info_dst.OSVERSION	Operating system version
Info_dst.FPGA_PRODUCTID	Product ID of the FPGA

Info_dst.TARGET	Returns the processor architecture of the platform
-----------------	----------------------------------------------------

The elements of the structure listed in the above table exist in Rex versions 01.01.020.

1.6.3.12 HWINFOONLINE

Provides information on the specified remote control.

Transfer parameters		Type	Description
interface			
address			Address of the PLC (see Using Online Functions)
Info_dest			Provides the name of the variable in which the information is written
Return parameters		Type	Description
			0 Function successful
			#0 Error code

The info_dst variable contains the following variables:

Variable:	Description
Info_dst.PLATFORM	Name of the platform, e.g. "C-IPC AMD LX800"
Info_dst.FPGAVER	The FPGA version
Info_dst.SERIAL	Serial number of the device
Info_dst.OSVERSION	Operating system version
Info_dst.FPGA_PRODUCTID	Product ID of FPGA
Info_dst.TARGET	Returns the process architecture of the platform

Requirements

This command is supported by Rexxx.dlm starting with version 01.01.024. Under Salamander it is supported starting with OS version 09.03.080 and 09.02.020 respectively.

1.6.3.13 KBHIT

Checks whether a key is available in the keyboard buffer.

Return parameters	Type	Description
		0 No key has been pressed
		≠0 A key has been pressed

1.6.3.14 LSLLINK

Links the loaded modules in a remote PLC.

Transfer parameters	Type	Description
interface		
address		Address of the PLC (see Using Online Functions)
Return parameters	Type	Description
		0 Function successful
		≠0 Error code

1.6.3.15 LSLLOADPRJ

Resets all loaded modules and optionally sets the project name in a remote PLC.

Transfer parameters	Type	Description
interface		
address		Address of the PLC (see Using Online Functions)
prjname		Project name
Return parameters	Type	Description
		0 Function successful
		≠0 Error code

1.6.3.16 LSLSEND

Transfers a LASAL project to a remote PLC.

Transfer parameters		Type	Description
interface			
address			Address of the PLC (see Using Online Functions)
Idx-file			Specifies the name of a project index file
Return parameters		Type	Description
			0 Function successful ≠0 Error code

1.6.3.17 LSLSENDMOD

Transfers a single LASAL module to a remote PLC.

Transfer parameters		Type	Description
interface			
address			Address of the PLC (see Using Online Functions)
lob-file			The file name of the LASAL object file
Return parameters		Type	Description
			0 Function successful ≠0 Error code

1.6.3.18 LSLSENDMODCLSNAM

This function transfers a single Lasal module to a remote PLC. In contrast to the LSLSENDMOD function, this function uses the class name and not the module ID to identify the module in the target system (Note: the module ID consists of the project name and class name). This function can be used to replace a class from a project whose name does not match the project name in the target system.

Transfer parameters		Type	Description
interface			

address		Address of the PLC (see Using Online Functions)	
lob-file		The file name of the LASAL object file	
Return parameters	Type	Description	
		0	Function successful
		≠0	Error code

1.6.3.19 MILLISLEEP

Pauses for the supplied number of milliseconds.

Transfer parameters	Type	Description	
milliseconds		Specifies the time in milliseconds for which to suspend execution	
Return parameters	Type	Description	
		0	Function successful
		≠0	Error code

1.6.3.20 OFFLINE

Shuts down an online connection to a remote PLC.

Transfer parameters	Type	Description	
online-descriptor		Online descriptor returned by the ONLINE() function	
Return parameters	Type	Description	
		0	Function successful
		≠0	Error code

1.6.3.21 ONLINE

Establishes an online connection to a remote PLC.

Transfer parameters	Type	Description	
Interface			

address		<p>Address of the PLC (see using online functions).</p> <p>Possible options in the address string. The options must be separated by a semicolon and must not contain spaces. These options can be used with all remote commands.</p> <p>"ROUTE=1" "ROUTE_PWD=pass" "ONLINE_PWD=pass"</p> <p>Example: ONLINE('IP','10.10.150.1;ROUTE=1;ROUTE_PWD=pass1;ONLINE_PWD=pass2')</p>
Return parameters	Type	Description
		If no error occurs the function returns the online-descriptor. Otherwise a negative error code is returned.

Use [OFFLINE\(\)](#) to shut down the connection. A program should always have a matching call to OFFLINE for each successful call to ONLINE to return any connection resources to the system.

When you use a TCP/IP connection, the remote PLC automatically closes the connection if no data is transferred within 30 s (newer operating systems use a timeout-value of 60 s). To maintain a TCP/IP connection, all functions that require a connection must be called (e.g. [GetCpuStatus\(\)](#)) at a regular time interval.

1.6.3.22 RESETSPS

Resets a remote PLC.

Transfer parameters	Type	Description	
interface			
address		Address of the PLC (see Using Online Functions)	
Return parameters	Type	Description	
		0	Function successful
		#0	Error code

1.6.3.23 REXEC

Executes a CLI command on a remote PLC.

Transfer parameters	Type	Description
interface		
address		Address of the PLC (see Using Online Functions)
Return parameters	Type	Description
		0 Function successful
		≠0 Error code



When successfully executed, the command is transferred and initiated to the remote PLC. The return code for this command is not the return code for the CLI command! This command also does not wait until the CLI command has finished; it is executed asynchronously.

1.6.3.24 RUNSPS

Starts the application on a remote PLC.

Transfer parameters	Type	Description
interface		
address		Address of the PLC (see Using Online Functions)
Return parameters	Type	Description
		0 Function successful
		≠0 Error code

1.6.3.25 RXFILE

Sends a file to the requesting control from a remote PLC connected via Ethernet.

Transfer parameters	Type	Description
interface		
address		Address of the PLC (see Using Online Functions)
Return parameters	Type	Description
		0 Function successful
		≠0 Error code

fname_scr		Provides the name of the source file (path+name in the remote control)				
fname_dst		Provides the name of the destination file (path+name of the file should have in the recipient control)				
Return parameters	Type	Description				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: right;">0</td><td>Function successful</td></tr> <tr> <td style="text-align: right;">#0</td><td>Error code</td></tr> </table>	0	Function successful	#0	Error code
0	Function successful					
#0	Error code					

1.6.3.26 SENDOS

Transfers and installs an operating system to a remote PLC. After the operating system is installed the PLC is rebooted.

Transfer parameters		Type	Description				
interface							
address			Address of the PLC (see Using Online Functions)				
osimage-file			Name of the file that contains the operating system image; this file has usually an extension of .BIN or .RTB				
Return parameters	Type	Description					
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: right;">0</td><td>Function successful</td></tr> <tr> <td style="text-align: right;">#0</td><td>Error code</td></tr> </table>	0	Function successful	#0	Error code	A failure of re-booting the PLC is not an error.
0	Function successful						
#0	Error code						

1.6.3.27 SERVERREAD

ServerRead returns a server value. Both local servers and servers from remote CPUs can be read. Servers can only be read when the CPU is in RUN.

Transfer parameters		Type	Description
interface			
address			Address of the PLC (see Using Online Functions)
servername			Name of the server
stem			Name of the variable to which the server value is written

Return parameters	Type	Description
		0 Function successful
		≠0 Error code

1.6.3.28 SERVERWRITE

With ServerWrite data can be written to a server. Both local servers and servers from remote CPUs can be written to. Servers can only be written when the CPU is in RUN.

Transfer parameters	Type	Description
interface		
address		Address of the PLC (see Using Online Functions)
servername		Name of the server
stem		New server value, variable or constant
Return parameters	Type	Description
		0 Function successful
		≠0 Error code

1.6.3.29 SETBREAKPOINT

Sets a breakpoint in the PLC.

Transfer parameters	Type	Description				
interface						
address		Address of the PLC (see Using Online Functions)				
nr		Breakpoint number (0-3)				
addr		Code address of the breakpoint				
len		Length of the breakpoint (only for data breakpoints)				
type		Type of the breakpoint <table border="1" style="margin-left: 20px;"> <tr> <td>0</td> <td>Instruction breakpoint</td> </tr> <tr> <td>1</td> <td>Instruction breakpoint AWL</td> </tr> </table>	0	Instruction breakpoint	1	Instruction breakpoint AWL
0	Instruction breakpoint					
1	Instruction breakpoint AWL					

		2 Data breakpoint write
		3 Data breakpoint write+read
stackOfs		Offset of the stack-area that is saved by the operating system when the CPU halts in a breakpoint
stackSize		Size of the stack-area that is saved by the operating system when the CPU halts in a breakpoint
cntr		Number of times the breakpoint condition must arrive before the CPU halts
code		Code for a conditional breakpoint
Return parameters	Type	Description
		0 Function successful ≠0 Error code

1.6.3.30 TXFILE

Transfers a file to a remote PLC.

Transfer parameters	Type	Description
interface		
address		Address of the PLC (see Using Online Functions)
fname_scr		Specifies the name of the source file
fname_dst		Specifies the name of the destination file e.g. the name of the file on the remote PLC. If the given file already exists on the remote PLC, its contents are destroyed.
Return parameters	Type	Description
		0 Function successful ≠0 Error code

1.6.4 Command Line Interface (CLI) Commands

You can send commands from a Rexx program to the Command Line Interface (CLI) either by issuing an expression to be executed as a command or by using the ADDRESS instruction with an environment name of SYSTEM.

After the CLI executes the command, the special RC variable is set to the return value of the CLI command. By convention, all CLI commands return a value of 0 when no errors occur.

Example

```
/* 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 Appendix A – Sample Rexx programs

VERSION

The VERSION program collects various system information and prints it on the screen. It is helpful to find out which version of Rexx is installed on the system.

```
/* 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

```

Automatic Program Start from Exchangeable Media

The following procedure demonstrates how to prepare a system to search for and execute a Rexx AUTOSTRT program on any exchangeable media (e.g. a USB stick). This procedure can be used to update the software on a PLC with minimal user intervention. The media need only be inserted where AUTOSTRT is located and the PLC rebooted. In this example the AUTOSTRT program copies files from the exchangeable media to a directory in the PLC.

First update the AUTOEXEC.LSL with the following entries:

```

SETENV Rexx_MACROS B:\;F:\;G:\;H:\;I:\;J:\;K:\;L:\;M:\;N:\;O:\;
Rexx AUTOSTRT
SETENV Rexx_MACROS C:\LSCMD

```

The first line sets the program search path to a list of drives with exchangeable media, the next line then tries to start the Rexx AUTOSTRT program from a directory specified in the previous step. This program is executed when an exchangeable storage device containing AUTOSTRT is inserted in the PLC during boot-up; otherwise an error message (program not found) is displayed. The last line resets the program search path.

The following is an example of an AUTOSTRT program:

```

/*-----
 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

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 '***      *****'
  Do Forever
    Nop
  End

  Return
```

1.6.6 Appendix B – Error Codes

The following table shows the error codes for REXX commands.

Errorcode	Description
0	No error
-200	Initialisation failed
-201	Invalid online type
-202	Address is already in use
-203	Out of memory
-204	No interface available
-205	Number of bytes in response is too small
-206	Timeout waiting for a cpu status
-207	Invalid address specified
-208	System error
-209	Maximum number of connections reached
-210	Invalid descriptor number
-211	Invalid parameter
-300	Could not create socket
-301	Connect failed
-302	Invalid ip address
-303	Tcp/ip error
-304	Timeout
-305	Buffer size is too small
-306	Connection closed
-307	Negative command acknowledge received
-308	Invalid response format
-309	No header found
-310	Logon failed

1.6.7 Appendix C – REXX How To

In this appendix, the basic functions of Rexx are explained in examples, to make life easier for programmers in future.

1.6.7.1 Function Call

Internal functions are marked with a label. If only one label marks the function, this function has access to all variables. If the keyword "Procedure" is behind the label, this function cannot access variables. With a "Procedure Expose" followed by a variable name, selected variables can be released for this function.

A function returning no value must be called via the "call" parameter. For the argument transfer in this case the brackets can be left away. See code example output 1 and 3. If the called function has a return parameter, it can be read from the variable "RESULT".



If a function calls a function and the subfunction needs access to a variable, this variable must be released for both functions.



If a function is called, the return value must be requested, especially on x86 CPUs. Otherwise unexpected behavior can occur.

```
a = 10
b = 50
erg = 0

/*Method Addition is called*/
call addition
say 'Output1: ' erg

/*Method Subtraction is called*/
erg = subtraction(b,a)
say 'Output2: ' erg

/*Method Multiplication is called*/
```

```
call multiplication a, b
say 'Output3: 'erg

/*
 * Methode Division is called*
 */
call division a, b
say 'Output4: 'RESULT

return

/*
 * This function can access all variables.*
 */
addition :
  erg = a + b
  return

/*
 * This function cannot access external variables.*
 */
subtraction : Procedure
  Parse Arg a, b
  return a - b

/*
 * This function can access the 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 Conditions and Loops

With conditions and loops it is possible to write a short or a long variant. If only one line should be executed, the short variant can be used, if more lines should be executed, the long variant must be used.

```
rc = 1

if (rc == 1) then
  Say 'Everything is OK.'
Else
  Say 'An error occurred.

if (rc <> 0) then do
  Say 'An error occurred.'
  Say 'Please restart!'
End
Else
  Say 'Everything is OK.

if (rc == 25) then
  Say 'The return value is 25'
Else do
  Say 'The return value is not 25'
  Say 'An error occurred'
End
```

In the short variant the “do” after the “then” is left away and so no “End” is needed. Both is needed in the long variant. In the example above, different variants and combinations are shown. The same is true for loops.

1.6.7.3 Online Functions

These are functions that establish a connection to a remote CPU. Basically, two parameters are needed here: interface and address. For each online function a connection to the remote CPU must be established by calling the function [ONLINE\(\)](#). If the online connection is no longer needed, the function [OFFLINE\(\)](#) should be called to release

the resources. “ONLINE” has an online descriptor, which is used as a parameter for the online function.

But the connection can also be established by directly transferring the interface type and the address to the online function. The advantage is, that you no longer have to call the functions “ONLINE” and [OFFLINE\(\)](#). The disadvantage is, that for each online function a new online connection is established and only released after a certain time. This needs a lot of resources, so you might run out of them.

It is recommended to always use online functions by calling the function [ONLINE\(\)](#).

The interface and address parameters can have the following values:

Interface	Address
CAN1 or CAN	<CAN station number> (0-31)
CAN2 or CAN	<CAN station number> (0-31)
IP1 or IP	<IP number> (dotted decimal format)
DESCR	<descriptor number return from ONLINE>



Some commands accept a local interface type (LOCAL). In this case the target is the local CPU and not the remote CPU. The address parameter for a LOCAL interface is any string (no empty string).

1.6.7.3.1 ONLINE

Establishes an online connection to a remote CPU.

Transfer parameters	Type	Description
interface		
address		Address of the PLC
Return parameters	Type	Description
		If no error occurs, the function returns the online descriptor. Otherwise it returns a negative error code.

1.6.7.3.2 OFFLINE

Closes an online connection to a remote CPU.

Transfer parameters	Type	Description
online-descriptor		Online descriptor returned by the function ONLINE
Return parameters	Type	Description
		0 Success ≠0 Error code

1.6.7.3.3 Example Function

```
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, 'cpuip' not
available('desthandle').'
end
else do
    say 'Online connection to CPU "'cpuip'"'
succeeded.'
end
return desthandle
```

1.6.7.4 Executing CLI Commands

For CLI commands there are differences, if they are executed on the local CPU or on a remote CPU. Additionally, there are differences, if they are executed on a x86 or an ARM CPU.

1.6.7.5 Address System cmd

This executes a CLI command on a local CPU. The shown command sets the variable "FILENAME" to "LONG".

If a CLI command is executed on the local CPU, NO "EXEC" must be in front of the command.

```
cmd = 'SETENV FILENAMES LONG'
Address System cmd
```

1.6.7.5.1 REXEC

This command executes a command on a remote CPU. If the local CPU is of the type "ARM", an "EXEC" must be in front of the command (cmdARM). If it is a "x86" CPU, no "EXEC" must be in front (cmdX86).

```
cmdX86 = 'SETENV FILENAMES LONG'
cmdARM = 'EXEC SETENV FILENAMES LONG'
rc = REXEC('INTERFACE', 'ADDRESS', cmdX86)
rc = REXEC('INTERFACE', 'ADDRESS', cmdARM)
```

Transfer parameters	Type	Description	
interface		The interface type LOCAL is supported	
address		Address of the PLC (see Using Online Variables)	
cmd		The command that should be executed on the remote CPU	
Return parameters	Type	Description	
		0 Success ≠0 Error code	

If executed successfully, the command is transferred and initialized on the remote CPU. The return code for this command is not the return code for the CLI command! This command does not wait, until the CLI command is finished. It is executed asynchronously.

1.6.7.5.2 SHOWENV

When calling the [SHOWENV\(\)](#) command on a x86 CPU, the variable name (e.g. FILENAMES) must be transferred. On a Salamander CPU everything is output, no matter whether with or without variable name.

1.6.7.5.3 Pipe Operator „>“

The pipe command can be used to redirect outputs e.g. to a file.

```

Parse Source . . filename
drive = LEFT(filename,2)
outfile = drive'\mac.txt'
cmd = 'ip >' outfile
Address System cmd
If rc \= 0 Then Do
  say 'Redirecting command IP to file FAILED'
End

```

In the example network settings are saved in a file. Then from it the IP, Mac address, etc. can be parsed.

This functionality is not supported under Salamander.

1.6.7.5.4 GETDATAD

Reads data from the application data area of a local or remote CPU.

```

/*Reading the system variable _WhoAmI in the place M02E8
 Possible values see lsl_st_kernel.h
 (e.g. #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	Type	Description
interface		The interface type LOCAL is supported
address		Address of the PLC (see Using Online Variables)
mem-addr		Memory address based on the start of the application data area
Length		Length of the searched memory; only values between 1 and 256 are allowed
Return parameters	Type	Description
Stem		The output is written to the stem of the compound variable. <stem>.len shows the length, <stem>.0 the first byte and so on.

1.6.7.6 File System Operations

In this part reading from a file and writing to a file is explained.

1.6.7.6.1 stream

This command offers many possibilities to execute file operations. Some of them are explained in the following.

Opening a File for Writing

This command opens a file and it is not overwritten, instead new contents is added behind the existing one resp. in a new. line.

Call STREAM 'C:\exampleTextFile.txt','C','OPEN WRITE APPEND'

If the file does not exist, it is created. If the contents of an existing file should be overwritten, “OPEN WRITE REPLACE” must be used instead of “OPEN WRITE APPEND”.

Closing a File

This command closes an open file.

rc = stream('C:\exampleTextFile.txt','C','CLOSE')

Checking whether a File/Folder Exists

This command checks whether the given folder exists. If not, it returns an empty string. A check is implemented in the example.

```
rc = stream('C:\exampleTextFile.txt', 'C', 'QUERY EXISTS')
if (rc <> '')
  say 'The file exists.'
Else
  say 'The file does not exist.'
```

Determining the Size of a File

This command returns the size of the file in bytes. If the file does not exist, an empty string is returned.

```
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

This function outputs the string in the given stream. So in our case this function is used to write in a file. For this the file must already be opened via the "stream" command. If this is not the case, the function returns an error.

```
string = 'This sentence should be written to the file.'
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
```

In the example we check, whether it worked. If not, the file is opened.

Transfer parameters	Type	Description	
stream		Stream where the string should be output	
string		String to be written to the stream	
line		Line number where the string should be written to the stream	
Return parameters	Type	Description	
		0	Writing successful
		1	Writing not successful

1.6.7.6.3 linein

In the shown case this function always reads the next line and saves it in the variable rc. When the end of the file is reached, it starts from the beginning. If no parameters are transferred to the function, it reads the inputs of the CLI interface.

```
rc = linein('C:\config.txt')
```

Transfer parameters	Type	Description	
Stream		Stream to be read	
Line		Line number to be read	
Count		Number of lines to be read	
Return parameters	Type	Description	
		This function returns Null, if an error occurs	

1.6.7.6.4 lines

This function returns the remaining number of lines in the stream.

```
if (lines('C:\config.txt') == 0)
  Say 'End of file reached'

if (lines('C:\config.txt') == 25)
  Say 'There are still 25 lines in the stream.'
```

```

if ((lines('C:\config.txt') == 1) | (lines('C:
\config.txt', 'N') == 1))
  Say 'There are still one or more lines in the stream.'

```

Transfer parameters		Type	Description
Stream			Defines the stream, of which to output the remaining number of lines
Option			Here you can choose if the exact number of lines ("C") or if lines still exist ("N") is shown. If no parameter is transferred, it returns either the exact number of lines or if still lines exist.
Return parameters		Type	Description
			See example

1.6.7.6.5 attrib

Changing file and folder attributes is realized via CLI commands.

```

cmd = 'attrib C:\exampleFolder -r'
rc = REXEC('Interface', 'Address', cmd)
Say rc
Address System cmd
say rc

```

This command must be written exactly like in the example above, otherwise it only works for x86 CPUs or for ARM CPUs.

Transfer parameters		Type	Description
path			Path to the folder or file for which the attributes should be changed
option			<p>Here the attribute to be changed is defined</p> <ul style="list-style-type: none"> + sets an attribute - removes an attribute a archive attribute h hidden attribute r read-only attribute s system attribute

1.6.7.7 REXX Error Codes

In this part the error codes are explained in more detail.

1.6.7.7.1 -307 – Negative command received

This error means, that an error occurred on the remote CPU. This error can occur e.g. during a copy procedure, if there is no more free memory on the remote CPU.

1.6.7.8 Useful Code Sections

1.6.7.8.1 Converting

Converting a string to upper/lower case

```
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 Reading and Processing

Reading and processing a file line by line

```
Parse Source . . filename  
drive = Left(filename,2)  
filePath = drive'\config.txt'  
/*Checking whether the file exists*/  
rc = stream(filePath, 'C', 'QUERY EXISTS')  
arrayLines.0 = ''  
arrayWords.0 = ''  
LinesCounter = 0  
WordsCounter = 0  
if (rc <> '') then do
```

```
/*Checking whether the file still has lines*/
do while (Lines(filePath,'c') > 0)
    /*Reading line from file*/
    rc = linein(filePath)
    Select
        /* Reading lines */
        when (word(rc,1) == 'Lines') then do
            arrayLines.LinesCounter = rc
            LinesCounter = LinesCounter + 1
        End
        /* Reading word */
        when (word(rc,1) == 'Words') then do
            arrayWords.WordsCounter = rc
            WordsCounter = WordsCounter + 1
        End
        otherwise do
        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 Creating Log File

Creating a log file

```
WarnCounter = 0  
ErrorCounter = 0  
logFile = ''  
/*With these variables the debug entries can be activated or deactivated*/  
/* 1 .. activates the debug entries  
any other value deactivates them*/  
debug = 0  
/*Determining drive from which the script is executed.*/  
Parse Source .. filename  
drive = Left(filename,2)  
drive = drive'\'  
/*Finding current date and creating log file*/  
actDate = DATE('s')  
logFile = drive'actDate'.log'  
Call STREAM logFile,'C','OPEN WRITE APPEND'  
call SetLogdataToFile ' - - - - - , '0'  
call SetLogdataToFile ' / \ | | / \ | \ / / , '0'  
call SetLogdataToFile ' | | | | | | / | | \ / , '0'  
call SetLogdataToFile ' \ | | | | | | / | | \ / , '0'  
call SetLogdataToFile ' - \ | | | | | | \ | | \ / , '0'
```



```

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 Creating a Path

Creating a path

```

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 Copying a Folder 1

Copying a folder with subfolder and contents on a local 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
        /*Inverted commas probably
because of file names with spaces*/
        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
    /*Changing the file
attributes to not write-protected, so that they can be
overwritten*/
    Address System 'attrib
    'cpcmddst' -r'
    End
    say 'Copying 'cpcmdsrc' to
    'cpcmddst
```

```

        cmd = 'COPY 'cpcmdsrc' 'cpcmddst
        Address System cmd
        if (rc <> 0) then do
            Say 'Copying 'cpcmdsrc'
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 Copying a Folder 2

Copying a folder with subfolder and contents from a local CPU to a 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
    End

```

```
do while (i <= list.0)
    /*Creating command and paths*/
    if (list.i.A_SUBDIR == 0) Then do
        /*Inverted commas probably because of
        file names with spaces*/
        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
            /*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
    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 Copying a Folder 3

Copying a folder with subfolder and contents from a remote CPU to a local 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
  /*Path to Dirlist*/
  SourceDirListFile = Source'\DirListName
  /*Create a new DirList on the remote CPU*/
  /*Selecting the command to create a DIRLIST, so that it is
  executed. If the CPU type of the terminal is not known, the
  command is executed with EXEC, like the example for ARM.*/
  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
    /*Copying Dirlist and creating destination folder*/
    rc = make_dir(Dest)
    if (rc == 0) then do
      rc = RXFILE('DESCR', desthandle,
      SourceDirListFile, Dest'\DirListName)
```

```
        /*Deleting the Dirlist file on the remote CPU,
as it has been copied and should not be copied again*/
        if (rc == 0) then do
            /*Selecting the command to delete a
DIRLIST, so that it is executed. If the CPU type of the terminal
is not known, the command is executed with EXEC, like the example
for ARM.*/
            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
                /*Parsing Dirlist and copying
files and subfolders*/
                do while
(LINES(Dest'\DirListName))
                rc =
LINEIN(Dest'\DirListName)
                if (CPUType == 'ARM') then
do
                /*Checking whether
this line is valid. It is valid, if it has 4 words (folder or
file names with spaces) and if the second word is not File(s),
because there is a line which also has 4 words*/
                if ((words(rc) == 4)
& (word(rc, 2) <> 'File(s)')) then do
                /*Checking
whether this PATH is not a folder, if yes the PATH for the file
```

```
is created, if no this function is called again for the
subfolder*/
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)
    /*Checking whether
    this line is valid. It is valid, if it has 5 words*/
    if (words(rc) == 5)
  then do
    /*Checking
    whether this PATH is not a folder, if yes the PATH for the file
    is created, if no this function is called again for the
    subfolder*/
    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
/*Deleting Dirlist.txt on the 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
End
Else do
    Say 'Creating the Dirlist.txt file
('SourceDirListFile') failed. Error code: 'rc
End
return rc
```

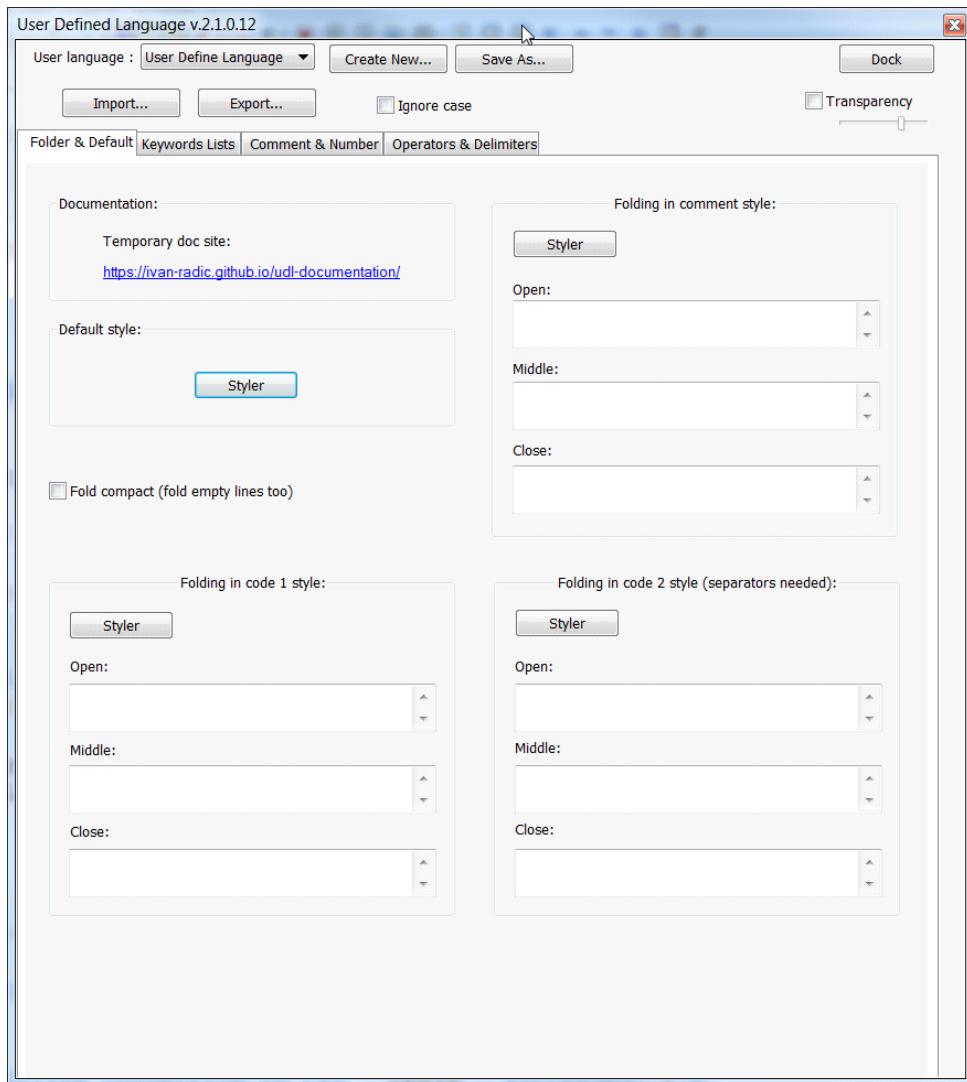
1.6.7.9 Notepad++ Highlighting

For adding code highlighting create a new text file, copy the text from [Code Highlighting](#) and save the file as “REXX.xml”. Then open Notepad++ and select “Define your language...” from the “Language” menu. Then a window opens, where you press “Import” and select the created “REXX.xml” file. Then restart Notepad++ and now in the “Language” menu you can select “REXX” below “Define your language...”.

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

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 say 'Online connecti
5521



1.6.7.10 Links

Some useful websites for REXX programming.

Kilowattsoftware

On this website nearly every standard REXX function is explained very well, also with examples. You just have to scroll through it a bit.

<http://www.kilowattsoftware.com/tutorial/rexx/default.htm>

Rexxinfo

Here, a lot of different functions are explained. Additionally, there is a directory with many other REXX tutorials.

<http://www.rexxinfo.org/html/functions.html> - REXX functions

<http://www.rexxinfo.org/html/instructions.html> - REXX instructions

<http://www.rexxinfo.org/html/rexxinfo1.html> - directory with REXX tutorials

Tutorialspoint

Also a good website with a tutorial for REXX. This website also offers a REXX development environment. Very helpful to test smaller parts like the parsing of string or similar things.

<https://www.tutorialspoint.com/rexx/index.htm> - tutorial

https://www.tutorialspoint.com/execute_rexx_online.php - development environment

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 What is LasalOS VNCcli?

LasalOS VNCcli is a VNC-client application. The VNC protocol is a simple protocol used for remote access of graphical user interfaces and is based on the remote framebuffer (RFB) concept. The protocol simply allows a server to update the frame buffer displayed on a viewer. Because it operates at the frame buffer level, it can be used with all operating or Windows systems and applications.

The VNC-client allows the user to work on any remote system, which runs a VNC server (Lasal OS, Windows, MacOS, Linux, Unix, ...).

VNC servers for Windows, Linux, Unix, and other systems are available free of charge on the following web pages: <http://www.realvnc.com> and <http://www.tightvnc.com>.

1.7.2 The Purpose of this Document

This document describes the LasalOS interface for the VNC Client application.

Requirements

LasalOS Requires LasalOS 1.1.3

Platform LasalOS VNCcli is available on the C-IPC

Limitations

The original application supports different color modes, including indexed palettes. Since the original pixel format for the C-IPC is 16-bit rgb565 (2 bytes per pixel, 5 bits red, 6 bits green, 5 bits blue), it is currently the only supported pixel format. This should not pose a problem since all VNC servers can support any pixel format.

Starting with Salamander 09.04.010

Salamander 09.02.150

Gecko 09.07.040

the color depths 8 bit, 16 bit and 32 bit are supported now, before only 16 bit.

Installation

- To use the VNC client with a C-IPC, copy the VNCCLI.DLM and ZLIB.DLM files to the C:\LSLSYS directory of the target system.

- Edit the AUTOEXEC.LSL file and insert the line 'VNCCLI START' to start the VNC client service at startup.

1.7.3 LASAL OS VNCcli-API Functions

The VNCCLI OS interface is available when the VNC client supports the user platform and the VNC client service is started by adding the command VNCCLI START to the autoexec.lsl file. The interface structure, the function prototypes and the constants used are defined in the lsl_st_vnc.h file.

Functions whose return parameter is a DINT have failed if they return a value greater than 0. See the paragraph [Error Codes](#) for a description of each error codes.

1.7.3.1 The OS Interface VNCCLI

```
TYPE
  OS_VNCCLI           : STRUCT
    udSize             : UDINT;
    udVersion          : UDINT;
    udVersionVNC       : UDINT;
    VNCInit            : pvoid;
    VNCExit             : pvoid;
    VNCConnect          : pvoid;
    VNCDisconnect       : pvoid;
    VNCDisconnectHandle : 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
```

The following is a short example of how this Interface is used:

Example

```

...
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 Function

The VNC client service must be able to communicate changes in the connection status to the corresponding LASAL CLASS object. If the VNC server disables the connection, for example, the LASAL object must be notified of the change. A global callback function must therefore be registered, which is called when a LASAL CLASS object must be notified of a status change.

For each VNC connection, the pThis pointer of a user class can be transferred to the VNCConnect function as parameter myThis. This pointer should point to the VNCClient application class in the LASAL CLASS project. The pThis parameter of the callback function is the same value as myThis. It can be assigned to a pointer to the user class. This allows the callback function to call the functions of the user class and access the member variables of the user class. See the example below.

Transfer parameters	Type	Description						
pThis	PVOID	This parameter is a pointer to the LASAL class object that should be notified of changes. This is the same pointer/value that is used as the myThis parameter of the vncconnect function.						
func	UDINT	<p>This parameter is a function code.</p> <table border="1"> <thead> <tr> <th>Symbol name</th> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>VNCMSG_CO_NNECT</td> <td>1</td> <td>Used each time a connection to a VNC server is established with a call to the VNCConnect() function.</td> </tr> </tbody> </table>	Symbol name	Value	Description	VNCMSG_CO_NNECT	1	Used each time a connection to a VNC server is established with a call to the VNCConnect() function.
Symbol name	Value	Description						
VNCMSG_CO_NNECT	1	Used each time a connection to a VNC server is established with a call to the VNCConnect() function.						

		VNCMSG_DISCONNECT	2	This function code notifies the LASAL object when a connection is terminated. It is usually triggered when VNCDisconnect() is called but is also executed when the server has terminated the connection or an error has occurred.
		VNCMSG_UPDATE	3	This function code is used to refresh the VNC message
param	UDINT	For each function code (func parameter) one or more parameter codes are defined:		
		VNCMSG_CONNECT		
		VNCMSG_CONNECT_STARTED	0	The connection sequence has started but a connection has not yet been established.
		VNCMSG_CONNECT_DONE	1	A connection to the specified host has been made. Events can now be sent and the remote screen shown.
		VNCMSG_CONNECT_FAILED	2	The connection to the host has failed; the _VNCHANDLE is now invalid.
		VNCMSG_CONNECT_AUTH_FAILED	3	The connection to the host was unsuccessful; authentication failed due to an invalid user/password combination The _VNCHANDLE is now invalid.
		VNCMSG_DISCONNECT		
		VNCMSG_DISCONNECT_STARTED	0	The disconnect sequence is started.
		VNCMSG_DISCONNECT_DONE	1	The disconnect sequence is complete The _VNCHANDLE is now invalid.
		VNCMSG_DISCONNECT_EXIT	2	The client has been disconnected because the VNC service client has been stopped (VNCExit or VNCReset has been called). The _VNCHANDLE is now invalid.
		VNCMSG_UPDATE		

		VNCMSG_UP DATE_SCREE N	0	The remote screen has changed and needs to be refreshed by calling VNCDraw() .
--	--	------------------------------	---	------------------------------------------------------------------------------------------------------

Prototype

```
FUNCTION __cdecl GLOBAL MyVNCCallback
VAR_INPUT
    pThis      : pvoid;
    func       : UDINT;
    param      : UDINT;
END_VAR;
```

Implementation Example

If `_VNCClient` is the LASAL class implementing the VNC client:

```
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;
(* Here the member function of the LASAL-class is called.*)
pClient^.VNCCallback(func, param);

END_FUNCTION //VIRTUAL GLOBAL _VncClient::VNCCallback
```

The `VNCCallback` member function of the class `_VncClient` may be implemented like this example:

Example of VNC Callback Function

```
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

NO FUNCTION POINTER!

An UDINT variable containing the size of the VNC client interface structure.

1.7.3.4 udVersion

NO FUNCTION POINTER!

An UDINT variable containing the DLL version of the VNC client DLL.

Currently only the LOW WORD is used to define the tripartite DLL version number (#1.#2.#3):

HIGH WORD				LOW WORD			
31-28	27-24	23-20	19-13	15-12	11-8	7-4	3-0
Unused (0)				#1	#2	#3	

Example

DLL version 1.1.1 correspond to the hex-value 16#00001101.

1.7.3.5 udVersionVNC

NO FUNCTION POINTER!

An UDINT variable containing the VNC protocol version number.

The higher word contains the main version number, the lower word the low version number.

Example

VNC protocol version 3.8 correspond to the hex-value 16#00030008.

1.7.3.6 VNCInit

Initializes the VNC client. This function must be called before any other functions of this interface.



It is important to specify a pointer to a callback function, as the LasalOS will notify the LASAL objects that are used to display the remote image using this callback function. Sending an NIL-pointer means, that VNC connection can never be established!

Transfer parameters	Type	Description
logStr	^CHAR	<p>This parameter is a string that defines a log setting. The log string is divided into three sections, which are separated by colons: "[filter]: [stream]:[log level]". The first section defines a module name filter '*' and will log messages from all modules; the second section defines an output-stream. The stream name 'file' writes the log messages to a file (the filename is defined with the second parameter of this function). Other stream names could consist of 'stderr', 'stdout' or 'kernel'. The third section defines the Log-Level (0: only information messages and errors are logged, 200: all messages including debug-messages are logged). A significant logStr would be: "**:kernel:30".</p> <div style="text-align: center;"> <p>Use 'kernel' as a log stream to write the VNC log messages into the system's log file c:\sysmsg\event00.log.</p> </div>
LogFileName	^CHAR	If the 'file' is defined as an output stream in the logStr parameter, a name for the log file must be provided or this parameter will be ignored.
PVNCCallback	PVOID	The OS must be able to communicate changes in a VNC session (Connect, Disconnect, Redraw, ...) to a LASAL Class object; the global callback function must be specified.

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCInit
VAR_INPUT
    LogStr      : ^CHAR;
    logFileName : ^CHAR;
    Callback    : pvoid;
END_VAR;
```

Log Levels

Log level	Message type	Description
0	error	Write only errors into the log.
30	Info	Write additional information into the logs.
100	Debug	Only needed for debugging.
150	Extended	Only needed for debugging.

1.7.3.7 VNCConnect

This function is used to establish a connection to the specified VNC server.

Transfer parameters	Type	Description
hostinfo	^CHAR	This parameter contains the server's host name. A port number can also be added after the host name (after a colon; e.g. 10.10.100.100:2300) in the VNC server doesn't use the standard VNC port 5900.
username	^CHAR	This parameter provides the user name needed for authentication. If the server doesn't require a user name, the parameter must be set to NIL.
password	^CHAR	The server requires the password for authentication. If the server does not require a user name, the parameter password must be set to NIL.
myThis	pVoid	This parameter is the THIS pointer of the LASAL CLASS object that handles the VNC connection. See also: description of the callback function.
prio	UDINT	To set the priority of the VNC client thread. Values from 1 to 13 are permitted. The default setting is 5. The parameter is only used with the RTK operating system.
dontfree	UDINT	Permissible values are 0 or 1. Value=1: Does not release the memory after a disconnect and attempts to use the memory again at the next connect. The parameter is only used with the RTK operating system.
Return parameters	Type	Description
RetVal	_VNCHANDLE	Returns a _VNCHANDLE, which identifies the connection. In case of an error 0 is returned.



The function returns a value immediately, however, having a connection handle does not mean that a connection has really been established. Each function call is asynchronous. Otherwise, the system would be blocked until the next connection has been established or a time-out occurs because the service is unavailable.

The callback function provides the information as to whether the connection could be established successfully or not.

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCConnect
VAR_INPUT
    hostinfo      : ^CHAR;
    username      : ^CHAR;
    password      : ^CHAR;
    myThis        : pvoid;
    prio          : UDINT;
    dontfree      : UDINT;
END_VAR
VAR_OUTPUT
    retval        : _VNCHANDLE;
END_VAR;
```

1.7.3.8 VNCDDisconnect

This function terminates a single connection to a specified VNC server.

Transfer parameters	Type	Description
hostinfo	^CHAR	<p>Server name.</p>  <p>The name provided must be exactly the same as the string used to initialize the connection with the VNCConnect() function (incl. port).</p>
		<p>If there are two or more connections to a VNC server, the first connection is terminated. In such a case, it is better to use the VNCDDisconnectHandle function. In addition, the string "all" can be used to disconnect all clients.</p>
Return parameters	Type	Description
RetVal	_VNCHANDLE	Operating system RTK: Returns the _VNCHANDLE of the disconnected connection. If an invalid string was provided or the disconnection has failed, the function will return 0. If the function

		was called using "all" as a parameter, the number of disconnected hosts is returned. Other operating systems: Always 0.
--	--	----------------------------------------------------------------------------------------------------------------------------

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCDisconnect
VAR_INPUT
    hostinfo      : ^CHAR;
END_VAR
VAR_OUTPUT
    retval       : _VNCHANDLE;
END_VAR;
```

1.7.3.9 VNCDisconnectHandle

This function terminates a single connection to a specified VNC server by providing its _VNCHANDLE.

Transfer parameters		Type	Description	
Return parameters		Type	Description	
hVNC		_VNCHANDLE	Connection handle returned by vncconnect	
retval		DINT	VNCHAN DLE	Function successful
			0	Invalid handle was provided or the function has failed due to another error

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCDisconnectHandle
VAR_INPUT
    hVNC      : _VNCHANDLE;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;
```

1.7.3.10 VNCDraw

This function is used to display the remote picture and is called from within the Draw function of the LASAL CLASS function only (e.g. a derived _MyIO class from the library System-Visualisation-LseRtk-UserInterface). Siehe auch VNCDraw2().

Transfer parameters	Type	Description	
hVNC	_VNCHANDLE	Connection handle returned by vncconnect	
clientRect	_ROOM	Rectangle client, contains the absolute screen coordinates used to draw the remote picture	
scroll_x	DINT	Scroll picture right/left	
scroll_y	DINT	Scroll picture up/down	
Return parameters	Type	Description	
RetVal	DINT	0 Function successful ≠0 Invalid handle was provided or the function has failed due to another error	

scroll_x and scroll_y are valid only, if the client rectangle (clientRect) is too small to display the full remote picture.

Prototype

```
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

This function is not available with the RTK operating system.

A fast drawing routine that determines what needs to be redrawn and copies the corresponding areas into the image buffer.

Transfer parameters	Type	Description	
_hVNC	_VNCHANDLE	Connection handle returned by vncconnect	

clientRect	^_ROOM	Pointer to the area on the screen where you want to draw (absolute positions in relation to the screen)
scroll_x	DINT	Move image left/right (in pixels)
scroll_y	DINT	Move image up/down (in pixels)
Return parameters	Type	Description
retval	DINT	0 Function successful 1 Invalid handle or function failed due to another error

scroll_x and scroll_y are only valid if the client rectangle (clientRect) is too small to display the complete remote image.

Prototype

```
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

This function disconnects ALL clients and clears the global call back function specified in [VNCInit\(\)](#). The [VNCInit\(\)](#) function must be recalled in order to use the VNC client service after calling [VNCExit\(\)](#).



To disconnect all clients without deleting the callback information, the [VNCDDisconnect\(„all“\)](#) function can be used.

Parameters

None

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCExit;
```

1.7.3.13 VNCForceRepaint

Calling this function triggers a refresh of the entire client area.

Transfer parameters	Type	Description	
_hVNC	_VNCHANDLE	Connection handle returned by vncconnect	
Return parameters	Type	Description	
retVal	DINT	0	Function successful
		#0	Invalid handle was provided or the function has failed due to another error

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCForceRepaint
VAR_INPUT
    _hVNC      : _VNCHANDLE;
END_VAR
VAR_OUTPUT
    retval     : DINT;
END_VAR;
```

1.7.3.14 VNCFreeBuffer

This function is not implemented.

Transfer parameters	Type	Description	
bufnum	UINT	Number of the buffer to be released (0: release all)	
Return parameters	Type	Description	
retval	DINT	0	Fuction successful
		1	Invalid handle or function failed for another reason

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCFreeBuffer
```

```

VAR_INPUT
    bufnum    : UINT;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;

```

1.7.3.15 VNCFreeOldSockets

This function is only available with the RTK operating system.

Sockets that are not released quickly enough are closed with xn_abort.

Transfer parameters		Type	Description	
Return parameters		Type	Description	
enable		UINT	0	Function successful
retval		DINT	1	Invalid handle or function failed for another reason

Prototype

```

FUNCTION __cdecl GLOBAL G_VNCFreeOldSockets
VAR_INPUT
    enable UINT;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;

```

1.7.3.16 VNCGetSize

This function retrieves the height and width of the remote desktop.

Transfer parameters		Type	Description	
Return parameters		Type	Description	
hVNC		_VNCHANDLE	0	Connection handle returned by vncconnect
width		^DINT	1	Pointer to a DINT
height		^DINT	2	Pointer to a DINT
retVal		DINT	0	Function successful

		#0 Invalid handle was provided or the function has failed due to another error
--	--	--------------------------------------------------------------------------------

Prototype

```
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

This function must be called when the PLC is reset. The pointer to the callback function is deleted. This is necessary, as the callback function is only valid if the CPU is in RUN_RAM or RUN_ROM status. The function also disconnects all clients by calling the [VNCExit\(\)](#).

(Normally this function does not have to be called from within a LASAL function; the OS does this automatically when reset.)

1.7.3.18 VNCSendHIDEEvent

This function sends touch/mouse events to the VNC server. The `MyIO::GetEvent()` function is an ideal location in the LASAL source code from which to call this function.

Transfer parameters	Type	Description
hVNC	_VNCHANDLE	Connection handle returned by <code>vncconnect</code>
eventtype	UDINT	<code>_EVENT_HIDMOVE</code> <code>_EVENT_HIDPRESS</code> or <code>_EVENT_HIDRELEASE</code>
button	UINT	button code (same as in the <code>_EVENT</code> structure)
x	INT	X position where the event has occurred
y	INT	Y position where the event has occurred
Return parameters	Type	Description

retval	DINT	0 Function successful ≠0 Invalid handle was provided or the function has failed due to another error
--------	------	---------------------------------------------------------------------------------------------------------

Prototype

```
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

This function sends keyboard events to the VNC server. The MyIO::GetEvent function is an ideal location in the LASAL source code from which to call this function.



If there are keys with the OS functions, such as arrow keys used change focus, it is practical to provide additional buttons or a menu that would allow the use of these keys as well.

Transfer parameters		Type	Description_
_hVNC	_VNCHANDLE		Connection handle returned by vncconnect
Return parameters		Type	Description
retval	DINT		0 Function successful ≠0 Invalid handle was provided or the function has failed due to another error

Prototype

```
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

This function is used to define or query whether the VNC client should send keyboard events to the VNC server or not.

Transfer parameters	Type	Description	
_hVNC	_VNCHANDLE	Connection handle returned by vncconnect	
set	DINT	0	get the value
		1	set the value
pSend	^DINT	A pointer to an integer where the value is stored	
Return parameters	Type	Description	
retval	DINT	0	Function successful
		≠0	Invalid handle was provided or the function has failed due to another error

Prototype

```
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

This function is used to define or query whether the VNC client should send pointer (mouse or touch screen) events to the VNC server or not.

Transfer parameters		Type	Description	
hVNC		_VNCHANDLE	Connection handle returned by vncconnect	
set		DINT	0 Get value 1 Set value	
pSend		^DINT	A pointer to an integer where the value is stored	
Return parameters		Type	Description	
retval		DINT	0 Function successful #0 Invalid handle or function has failed due to another error	

Prototype

```
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

This function lets the VNC client thread sleep in order to reduce the CPU load.

Transfer parameters		Type	Description	
_hVNC		_VNCHANDLE	Connection handle returned by vncconnect	
sstate		UINT	0 Continue processing messages 1 Send thread to sleep	
kalive		UINT	0 No event is generated	

		1 In sleep mode, an event is sent to the VNC server every 10 s to prevent a timeout (default 3600 s)								
Return parameters	Type	Description								
retval	DINT	<p>Operating system RTK</p> <table border="1"> <tr> <td>sstate</td><td>Fuction successful</td></tr> <tr> <td></td><td>1 Invalid handle or function failed due to another error</td></tr> </table> <p>Other operating systems</p> <table border="1"> <tr> <td>0</td><td>Fuction successful</td></tr> <tr> <td>1</td><td>Invalid handle or function failed due to another error</td></tr> </table>	sstate	Fuction successful		1 Invalid handle or function failed due to another error	0	Fuction successful	1	Invalid handle or function failed due to another error
sstate	Fuction successful									
	1 Invalid handle or function failed due to another error									
0	Fuction successful									
1	Invalid handle or function failed due to another error									

Prototype

```
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

This function specifies the timeout for establishing the connection. It is available for the RTK operating system from version 1.1.27 of the VNC client DLL.

Transfer parameters	Type	Description
TimeOut	UDINT	Time in seconds
Return parameters	Type	Description
retval	DINT	Always 0

Prototype

```
FUNCTION __CDECL GLOBAL G_ VNCSetConnectTimeout
VAR_INPUT
    Timeout: UDINT;
END_VAR
```

```
VAR_OUTPUT
  retval    : DINT;
END_VAR;
```

1.7.3.24 VNCSetKaValues

This function is used to configure the sending of keep-alive packages. Keep-alive packets are sent from the VNC client to the VNC server after a communication error. If no more packets are exchanged on a connection, keep-alive packets are sent to check whether the communication partner is still connected.

Transfer parameters		Type	Description				
_hVNC		_VNCHANDLE	Connection handle returned by vncconnect				
usInterval		UINT	Waiting time in seconds until the first keep-alive packet is sent after a communication error. Enter 0 to not send any keep-alive packets.				
usRetry		UINT	Time in seconds between keep-alive packages				
usTimeout		UINT	Keep-alive packets must be answered within this time in seconds. An error is then set and the connection is interrupted.				
Return parameters		Type	Description				
retval		DINT	<table border="1"> <tr> <td>0</td><td>Function successful</td></tr> <tr> <td>1</td><td>Invalid handle or function failed due to another error</td></tr> </table>	0	Function successful	1	Invalid handle or function failed due to another error
0	Function successful						
1	Invalid handle or function failed due to another error						

Prototype

```
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

This function is not available with the RTK operating system.

This function can be used to configure the VNC client.



The values for VNCCLI_OPTION_ENCODING_FORMAT are only adopted when the connection is established.

Transfer parameters	Type	Description	
optionId	DINT	ID of the parameter to be set (see table)	
optionAddr	^void	Address where the parameter value is stored	
optionSize	DINT	Size of the parameter value in bytes (see table)	
Return parameters	Type	Description	
retval	DINT	0 Function successful -1 Invalid optionAddr or invalid optionSize or optionId unknown	

Prototype

```
FUNCTION __cdecl GLOBAL G_ VNCSetOptions
VAR_INPUT
  optionId      : DINT;
  optionAddr    : ^void;
  optionSize    : DINT;
END_VAR
VAR_OUTPUT
  retval        : DINT;
END_VAR;
```

Possible Options

optionId	optionSize	Description	
VNCCLI_OPTION_USE_REMOTE_CURSOR (1)	4	0	Remote cursor is not displayed locally
		1	Remote cursor is displayed locally

VNCCLI_OPTION_COLOR_DEPTH (2)	4	Permitted values are only 16 and 24.												
VNCCLI_OPTION_ENCODING_FORMAT (3)	Length of the ASCII string plus 1	<p>The option values must be entered as ASCII strings.</p> <ul style="list-style-type: none"> • Tight • ZRLE • Ultra • Hextile • Zlib • CoRRE • RRE • Raw 												
VNCCLI_OPTION_CLEAR_SCREEN_AT_CONNECT (4)	4	<p>If inactive (set to 0 before connecting to the VNC server), the screen is not cleared when the connection is established. The last visible image is displayed until the first update is received from the VNC server.</p> <p>If active (set to 1 before connecting to the VNC server), the screen is cleared when the connection is established. A black screen is displayed until the first update is received from the VNC server.</p>												
VNCCLI_OPTION_SCALING_FACTOR (5)	4	<p>Scaling factor for the VNC server.</p> <table border="1"> <tr><td>0</td><td>No scaling</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">etc.</td></tr> </table> <p>Important Notes:</p> <p>The scaling factor must be set before the connection is established.</p> <p>The scaling factor is only supported from OS versions Salamander 09.02.163 and 09.04.023.</p> <p>The scaling factor is not supported by all VNC servers. VNC servers from SIGMATEK support the scaling factor.</p>	0	No scaling	1	1:1 (100 %)	2	1:2 (50 %)	3	1:3 (33 %)	4	1:4 (25 %)	etc.	
0	No scaling													
1	1:1 (100 %)													
2	1:2 (50 %)													
3	1:3 (33 %)													
4	1:4 (25 %)													
etc.														
VNCCLI_OPTION_USE_COLOR_DEPTH_FROM_SERVER (6)	4	If this option is set before the connection is established, the server's default color depth is used.												

1.7.3.26 VNCSetTaskDelayTime

This function is used to set the time by which the VNC client thread is blocked in each run.

Transfer parameters		Type	Description	
_hVNC	_VNCHANDLE	Connection handle returned by vncconnect		
delayTime	UINT	Time by which the VNC client thread is blocked in each run, in ms		
Return parameters		Type	Description	
retval	DINT	0 Function successful 1 Invalid handle or function failed due to another error		

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCSetTaskDelayTime
VAR_INPUT
  _hVNC      : _VNCHANDLE;
  delayTime  : UINT;
END_VAR
VAR_OUTPUT
  retval     : DINT;
END_VAR;
```

1.7.3.27 Error messages in Event00.log file

The VNC Client writes error and info messages into the Event00.log file.

Error message	Log level	Description
logParams: %s	Info	Logging parameter initialization
VNC Client initialized	Info	The VNC Client is now initialized
VNC client tries to set up a connection to host: %s (LASAL-Viewer: 0x%p)	Info	VNC Client will now start the connection
VNC client - disconnect <%s>	Info	VNC Client disconnected
VNC-Client service started: <%s> : <%s> : 0x%p	Error	More information about the VNC Client status
VNC Client initialization failed: not enough memory!	Error	Not enough OSHEAP for the VNC Client
VNCConnect failed : Couldn't create VNC-Viewer-thread!!	Error	Could not start VNC Viewer Thread
VNCConnect failed : VNC Client not initialized!	Error	Error in VNC Client initialization phase

VNCDisconnect for host <%s> failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCGetSize for handle 0x%08x failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCSendKeyboardEvents for handle 0x%08x failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCSendPointerEvents for handle 0x%08x failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCSendHIDEEvent for handle 0x%08x failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCSetCliSleep for handle 0x%08x failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCFreeBuffer failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCSetTaskDelayTime for handle 0x%08x failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCFreeOldSockets failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCDraw for handle 0x%08x failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
VNCForceRepaint for handle 0x%08x failed: No pViewManager!	Error	Internal pViewManager thread couldn't be started
EXCEPTION WHILE COPYING RECTS	Error	Error while copying image information to the pixel buffer
DIBSectionBuffer::recreateBuffer failed: NOT ENOUGH MEMORY for Pixelbuffer (%i Bytes)	Error	Not enough memory for pixel data. Increase OSHEAP
refresh palette called for truecolour DIB	Info	The color palette was reloaded
unable to connect to %s (%s)	Info	The connection couldn't be established
Connected to VNC-Server on host %s.	Info	VNC Client has been connected to server
Authentication failure : %s : reason: %s	Info	Error in authentication
Disconnecting host <%s>, Handle: %p	Info	The connection was closed
No such host (%s)	Info	Could not find host
stop all: No connections	Info	Currently there are no connections
stop all: disconnected %i hosts	Info	Number of disconnected hosts

SetCliSleep state:%d, keepalive:%d	Info	Option SetCliSleep was changed
SetTaskDelayTime %d	Info	Option SetTaskDelayTime was changed
FreeOldSockets %s	Info	Old sockets will be deleted from the CLOSED_WAIT list
CViewThread::VNCGetSize: VNC_NO_ATTACHED_VIEW	Error	No view connected to the VNC Client
CViewThread::VNCGetSize: VNC_NO_ATTACHED_VIEW	Error	No view connected to the VNC Client
CViewThread::VNCSendHIDEEvent: VNC_NO_ATTACHED_VIEW	Error	No view connected to the VNC Client
CViewThread::VNCDraw: VNC_NO_ATTACHED_VIEW	Error	No view connected to the VNC Client
CViewThread::VNCForceRepaint: VNC_NO_ATTACHED_VIEW	Error	No view connected to the VNC Client
CViewThread::VNCSendKeyboardEvents: VNC_NO_ATTACHED_VIEW	Error	No view connected to the VNC Client
CViewThread::VNCSendPointerEvents: VNC_NO_ATTACHED_VIEW	Error	No view connected to the VNC Client
RFB Exception in CViewThread::run(): %s	Error	An internal error has been occurred
CViewThread::run() : inner rdr::Exception: %s	Error	An internal error has been occurred
Failed to create CViewThread for host <%s>	Error	Could not create CViewThread
closing - %s	Info	Closing connection
CView::CView() : FAILED TO CREATE DIBSectionBuffer (buffer for remote-picture data)	Error	Cannot create Pixel buffer
%s: LockBuffer FAILED -> Closing	Error	Cannot lock pixel buffer
getUserPasswd failed: missing username or password.	Error	The username or password are incorrect

1.7.3.28 Error Codes VNC Client

Functions whose return parameter is a DINT have failed if they return a value greater than 0.

Symbol name	Value	Description
VNC_OK	0	Function successful. No errors.
VNC_NO_VIEWMAN GER	1	The main VNC client thread has been terminated unexpectedly.

VNC_NO_THREAD	2	The thread handling this connection has been terminated unexpectedly.
VNC_NO_BUFFER	3	The remote pixel buffer is not yet available. Functions: VNCGetSize
VNC_NO_ATTACHED_VIEW	4	The object for this connection has been terminated unexpectedly.
VNC_FAILED_BUFFER_LOCK	5	Currently unused
VNC_FAILED_BUFFER_UNLOCK	6	An error has occurred while signaling the remote pixel buffer flag.
VNC_INIT_FAILED	7	Not enough memory to initialize the VNC client. Function: VNCInit()
VNC_NOT_DRAWN	8	This error can occur if the VNC client didn't have enough memory to allocate the pixel buffer to get the remote picture.

1.7.4 Command Line Interface (CLI) Commands

The VNC client service can be started or stopped manually with the VNCCLI command:

Syntax

```
vnccli start | stop | version
```

Example

```
vnccli start
```

Starts the VNC-client service.

```
vnccli stop
```

Stops the VNC-client service.

```
vnccli version
```

Shows the DLL-version and the VNC-protocol-version numbers.

1.8 VNCsvr

1.8.1 What is LasalOS VNCsvr?

LasalOS VNCsvr is a VNC server application for the LasalOS. The VNC protocol is a simple protocol used for remote access of graphical user interfaces and is based on the remote frame buffer (RFB) concept. The protocol simply allows a server to update the frame buffer displayed on a viewer. Because it works at the frame buffer level, it can be used with all operating or Windows systems and applications.

The VNC server allows you to have remote access to your C-IPC from any system that supports the use of a standard VNC client (LasalOS, Windows, MacOS, Linux, Unix, ...).

VNC clients for Windows, Linux, Unix, and other systems are available free of charge on the following web pages: <http://www.realvnc.com> and <http://www.tightvnc.com>.

1.8.2 The Purpose of this Document

This document describes the LasalOS interface for the VNC server.

Requirements

LasalOS Requires LasalOS 1.1.3

Platform LasalOS VNCsvr is available on the C-IPC, ETV and EDGE

Installation

- To use the VNC server with a C-IPC, copy the VNCSV.R.DLM and ZLIB.DLM files to the C:\LSLSYS directory of the target system.
- Copy the file VNCPWD.TXT into the directory C:\ to the target system. The password stored in the file is encrypted.
- Edit the AUTOEXEC.LSL and insert the line 'VNCSV START' to start the VNC server service when the C-IPC boots up.

Terminals with Edge PLC are configured with a low OS heap (memory, reserved for the OS) setting. If the VNC is used, it is recommended that the OS heap be increased. (E.g. 10000 byte.)

Sample: SET OSHEAP 10000

- The default password of your VNC-server is 'SIGMATEK' – to change this password use the following CLI-command VNCSV SETPASS sigmatek <your new password> The VNC server must have been initialized with the VNCSV START command before setting a new password. (This can be done either through the [AUTOEXEC.LSL](#) or by entering the command in the CLI).
- After calling 'VNCSV START', the VNC server is ready for use and can be configured in the CLI. The OS- interface functions are only necessary if you want to provide a user-friendly interface.



The VNC server can be stopped by simply entering 'VNCSVR STOP' in the CLI.

- The repeater function is configured via the vncsvr.cfg file. If this file is available (in the folder C:\LsISys\ of the control) and all required entries (see [Configuring the Repeater](#)) are valid, the repeater function is started with the VNC server [automatically](#).

1.8.3 LASAL OS VNCsvr-API Functions

The VNCSVR OS interface is available when the VNC server supports user platform and the service is started by adding the VNCSVR START to the AUTOEXEC.LSL. The interface structure, the function prototypes and the constants used are defined in the lsl_st_vnc.h file.

1.8.3.1 The OS Interface VNCSVR

```
TYPE
  OS_VNCSVR
    udSize : STRUCT;
    udVersion : UDINT;
    udVersionVNC : UDINT;
    VNCSVRInit : 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;
    VNCSVRAlwaysShared : pvoid;
    VNCSVRMaxConnections : pvoid;
    VNCSVRClientWaitTimeMillis : pvoid;
    VNCSVRCompareFB : pvoid;
    VNCSVRSetDisplayOffset : pvoid;
    VNCSVRUpdateRect : pvoid;
    VNCSVRInitialized : pvoid;
    VNCSVRConnectedClients : pvoid;
    VNCSVRReserved0 : pvoid;
    VNCSVRReserved1 : pvoid;
    VNCSVRDisableSharedMemory : pvoid;
```

```

VNCsvrEnumConnections2           : pvoid;
VNCsvrConnectedClients2         : pvoid;
END_STRUCT;
END_TYPE

```

The following is a short example of how to use this interface.

Example

```

...
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

This is a simple UDINT variable containing the size of the interface structure.

1.8.3.3 udVersion

This is a simple UDINT variable containing the DLL-Version of the VNC-server-DLL.

Currently only the lower WORD is used to define the tripartite DLL version number (#1.#2.#3):

HIGH WORD				LOW WORD			
31-28	27-24	23-20	19-13	15-12	11-8	7-4	3-0
Unused (0)				#1	#2	#3	

Example

DLL version 1.1.1 corresponds to the hex-value 16#00001101.

1.8.3.4 udVersionVNC

This is a simple UDINT variable containing the VNC protocol version number. The higher word contains the main version number, the lower word the low version number.

Example

VNC protocol version 3.8 correspond to the hex-value 16#00030008.

1.8.3.5 VNCSVRAcceptKeyboardEvents

With this function, whether the VNC server allows remote keyboard access can be specified. The default value is true.

Transfer parameters		Type	Description				
set	DINT		Used to set or retrieve a value <table border="1" style="margin-left: 20px;"> <tr> <td>0</td><td>Retrieves value</td></tr> <tr> <td>1</td><td>Sets value</td></tr> </table>	0	Retrieves value	1	Sets value
0	Retrieves value						
1	Sets value						
pValue	^DINT		Pointer to an integer where the desired value is/should be stored				
Return parameters		Type	Description				
RetVal	DINT		0 Function successful				

See also: [VNC_SVR_ACCEPT_KEYBOARD](#)

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCSVRAcceptKeyboardEvents
VAR_INPUT
    set      : DINT;
    pAccept  : ^DINT;
END_VAR
VAR_OUTPUT
    retval   : DINT;
END_VAR;
```

1.8.3.6 VNCSVRAcceptPointerEvents

With this function, whether the VNC server allows remote device access (Touch screen & Mouse) can be specified. The default value is true.

Transfer parameters	Type	Description	
set	DINT	Used to set or retrieve a value	
		0 Retrieves value	
		1 Sets value	
pAccept	^DINT	Pointer to an integer where the desired value is/should be stored	
Return parameters	Type	Description	
retVal	DINT	0	Function successful

See also: [VNC_SVR_ACCEPT_POINTER](#)

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCSVRAcceptPointerEvents
VAR_INPUT
    set          : DINT;
    pAccept      : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.7 VNCSVRAlwaysShared

This function defines whether the server always allows shared connections. If it is set to 1, the server always allows shared connections regardless what is defined by the client. The default setting is 0.

Transfer parameters	Type	Description	
set	DINT	Used to set or retrieve a value	
		0 Retrieves value	
		1 Sets value	

pValue	^DINT	Pointer to an integer where the desired value is/should be stored
Return parameters	Type	Description
retval	DINT	0 Function successful

See also: [VNC_SVR_ALWAYS_SHARED](#)

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCVRAlwaysShared
VAR_INPUT
    set          : DINT;
    pValue       : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.8 VNCVRClientWaitTimeMillis

This property defines how long (in milliseconds) the server waits for a client, which does not respond; the default value is 20000.

Transfer parameters	Type	Description				
set	DINT	Used to set or retrieve a value <table border="1" style="margin-left: 20px;"> <tr> <td>0</td><td>Retrieves value</td></tr> <tr> <td>1</td><td>Sets value</td></tr> </table>	0	Retrieves value	1	Sets value
0	Retrieves value					
1	Sets value					
pValue	^DINT	Pointer to an integer where the desired value is/should be stored				
Return parameters	Type	Description				
retval	DINT	0 Function successful				

See also: [VNC_SVR_WAIT_FOR_CLIENT](#)

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCVRClientWaitTimeMillis
```

```

VAR_INPUT
    set          : DINT;
    pValue       : ^DINT;
END_VAR
VAR_OUTPUT
    retval       : DINT;
END_VAR;

```

1.8.3.9 VNCSVRCCompareFB

This property defines, whether the server performs a pixel comparison on the frame buffer to reduce unnecessary updates (1) or not (0). The default value is 0.

Transfer parameters	Type	Description	
set	DINT	Used to set or retrieve a value	
		0	Retrieves value
pValue	^DINT	1 Sets value	
		Pointer to an integer where the desired value is/should be stored	
Return parameters	Type	Description	
retVal	DINT	0 Function successful	

See also: [VNC SVR COMPARE FB](#)

Prototype

```

FUNCTION __cdecl GLOBAL G_VNCSVRCCompareFB
VAR_INPUT
    set          : DINT;
    pValue       : ^DINT;
END_VAR
VAR_OUTPUT
    retval       : DINT;
END_VAR;

```

1.8.3.10 VNCSVRDisconnectAllClients

This function disconnects all VNC clients from the server but the server remains active.

Return parameters	Type	Description
-------------------	------	-------------

retVal	DINT	0 Function successful
--------	------	-----------------------

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCVRDisconnectAllClients
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.11 VNCVRDisconnectClient

This function disconnects a specified client.

Transfer parameters	Type	Description
strClient	^CHAR	Client to be disconnected (e.g.: "192.168.10.14::3601"). One of the strings returned in the enumeration callback function can be used.
Return parameters	Type	Description
retval	DINT	0 Function successful

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCVRDisconnectClient
VAR_INPUT
    strClient      : ^CHAR;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.12 VNCVRConnectedClients

This function returns the number of clients connected to the VNC server.

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCVRConnectedClients
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.13 [VNCSVRCConnectedClients2](#)

This function is the same as the [VNCSVRCConnectedClients\(\)](#) function with the additional parameter `listenerPort`. This parameter allows to only consider connections established via the given port of the server (the port number of the server is given in the environment variable [VNC_SVR_PORT](#) resp. [VNC_SVR_PORT2](#)).

Transfer parameters	Type	Description
<code>listenerPort</code>	DINT	Defines the port number of the server. If this parameter is > 0, only the connections established via this port are considered. The value 0 means, that all connections are considered.
Return parameters	Type	Description
<code>retval</code>	DINT	

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCSVRCConnectedClients2
VAR_INPUT
    listenerPort      : DINT;
END_VAR
VAR_OUTPUT
    retval           : DINT;
END_VAR;
```

1.8.3.14 [VNCSVRDisconnectClientsOnNonsharedConnection](#)

Use this function to set or retrieve the property `DisconnectClientsOnNonsharedConnection`. If this property is set to 1, the server will disconnect all other clients when a client connects that doesn't have 'Shared connection' option set. If the property is set to 0, only clients that have the 'Shared connection' option set are allowed to connect and clients with the option 'non shared connection' are refused.

Transfer parameters	Type	Description				
<code>set</code>	DINT	Used to set or retrieve a value <table border="1" data-bbox="487 1134 1036 1229"> <tr> <td>0</td> <td>Retrieves value</td> </tr> <tr> <td>1</td> <td>Sets value</td> </tr> </table>	0	Retrieves value	1	Sets value
0	Retrieves value					
1	Sets value					
<code>pDisconnect</code>	<code>^DINT</code>	Pointer to an integer where the desired value is/should be stored				
Return parameters	Type	Description				

retval	DINT	0 Function successful
--------	------	-----------------------

See also: [VNC_SVR_DISCONNECT_ON_NONSHARED](#)

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCVRDisconnectClientsOnNonsharedConnection
VAR_INPUT
    set          : DINT;
    pDisconnect : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.15 VNCVRDisableSharedMemory

This function can prevent the Full Screen update from the shared memory, if more than 75 % of the screen content has changed.

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCVRDisableSharedMemory
VAR_INPUT
    set          : DINT;
    pValue      : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

This feature is supported from interface version 01.01.037.

This feature is only supported in Salamander.



1.8.3.16 VNCSVREnumConnections

The callback function provided is called once for each VNC client connected.

The pThis parameter is provided so that a member function can be called from the global callback function. This can be used as a pointer to a class that can call the member function. After the last connection, the callback function is called with a NULL pointer as the hostinfo parameter to indicate the end of the list.

Transfer parameters	Type	Description	
pThis	PVOID	Pointer to the LASAL Class object, which handles this connection	
callback	PVOID	Pointer to a global enumeration callback function	
Return parameters	Type	Description	
retval	DINT	0	Function successful

Callback-Prototype

```
FUNCTION __cdecl GLOBAL VNCSVREnumConnections
VAR_INPUT
    pThis      : pvoid; (* arbitrary Pointer - e.g. a LASAL-class *)
    hostinfo   : ^CHAR; (* "IPAddr::Port" e.g. 192.168.10.15::7600 *)
END_VAR;
```

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCSVREnumConnections
VAR_INPUT
    pThis      : pvoid;
    Callback   : pvoid;
END_VAR
VAR_OUTPUT
    retval     : DINT;
END_VAR;
```

1.8.3.17 VNCSVREnumConnections2

This function is the same as the [VNCSVREnumConnections\(\)](#) function with the additional parameter listenerPort. This parameter allows to only list connections established via the given port of the server (the port number of the server is given in the environmental variable VNC_SVR_PORT resp. VNC_SVR_PORT2).

Transfer parameters	Type	Description

pThis	PVOID	Pointer to the LASAL Class object handling this connection
Callback	PVOID	Pointer to a global enumeration callback function
listenerPort	DINT	Defines the port number of the server. If this parameter is > 0, only the connections established via this port are listed. The value 0 means, that all connections are listed.
Return parameters	Type	Description
retval	DINT	0 Function successful

Callback Prototype

```
FUNCTION __cdecl GLOBAL VNCSVREnumConnections
VAR_INPUT
    pThis           : PVOID; (* beliebige Pointer - z.B. auf eine LASAL-Klasse *)
    hostinfo        : ^CHAR; (* "IPAddr::Port" e.g. 192.168.10.15::7600 *)
END_VAR;
```

Prototype

```
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

This function disconnects all VNC clients and exits the VNC server.

Parameters

None

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCSVRExit;
```

1.8.3.19 VNCSVRFullRefreshCycle

With this function, the time intervals at which the VNC server to do a full screen update can be specified. The default value is 2 seconds; 0 disables this feature.

Transfer parameters	Type	Description				
set	DINT	Used to set or retrieve a value <table border="1" style="margin-left: 20px;"> <tr> <td>0</td><td>Retrieves value</td></tr> <tr> <td>1</td><td>Sets value</td></tr> </table>	0	Retrieves value	1	Sets value
0	Retrieves value					
1	Sets value					
pSeconds	^DINT	Pointer to an integer where the desired value is/should be stored				
Return parameters	Type	Description				
retVal	DINT	0 Function successful				

See also: [VNC_SVR_FULL_UPDATE](#)

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCSVRFullRefreshCycle
VAR_INPUT
    set      : DINT;
    pSeconds : ^DINT;
END_VAR
VAR_OUTPUT
    retval   : DINT;
END_VAR;
```

1.8.3.20 VNCSVRInit

This function initializes the VNC server. Because this function is called automatically when the VNC server is started with the CLI command vncsvr start, it is manually called only if the VNC server was stopped using the [VNCSVRExit\(\)](#) function.

Transfer parameters	Type	Description
logStr	^CHAR	This parameter is a string that defines a log setting . The log string is divided into three sections, which are separated by colons: "[filter]:[stream]:[log level]". The string in the first section defines a module name filter à ** logs all Messages from all modules (keep ** for normal use.).

		<p>The second section defines an output-stream; 'file' writes the log messages to a file (the filename is defined with the second parameter of this function). Other stream names could consist of 'stderr', 'stdout' 'kernel_svr'.</p> <p>The third section defines the log level . An applicable logStr would be: ".*.kernel_svr:30".</p> <div style="text-align: center;">  </div> <p>Use 'kernel_svr' as log-stream to write the VNC-log-messages into the system-log file c:\sysmsg\event00.log.</p>
LogFileName	^CHAR	<p>To add an extra log file for the VNC server, "file" has to be defined as an output stream in the logStr parameter and a name for the log file provided.</p> <p>Example: For a vncsvr.log log file, the server must be initialized the server as follows:</p> <pre>VNCSVRInit("*.file:30", "vncsvr.log")</pre>

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCVRInit
VAR_INPUT
    logStr      : ^CHAR;
    logFileName : ^CHAR;
END_VAR;
```

Log Levels

Log level	Message type	Description
0	error	Write only errors into the log.
30	info	Write some more information into the logs.
100	debug	Only needed for debugging.
150	extended	Only needed for debugging.

1.8.3.21 VNCSVRIinitialized

This function returns the value 1 when the VNC server is running, otherwise 0.

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCSVRIinitialized
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.22 VNCSVRMaxConnections

This function defines, how many clients are allowed to connect to the server simultaneously; the default value is 4.

Transfer parameters	Type	Description	
set	DINT	Used to set or retrieve a value	
		0	Retrieves value
		1	Sets value
pValue	^DINT	Pointer to an integer where the desired value is/should be stored	
Return parameters	Type	Description	
retVal	DINT	0	Function successful

See also: [VNC_SVR_MAX_CONNECTIONS](#)

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCSVRMaxConnections
VAR_INPUT
    set      : DINT;
    pValue   : ^DINT;
END_VAR
VAR_OUTPUT
    retval   : DINT;
END_VAR;
```

1.8.3.23 VNCSVRNeverShared

This property defines whether the server allows shared connections. If it is set to 1, the server does not allow shared connections regardless what is defined by the client. The default setting is 0.

Transfer parameters	Type	Description				
set		Used to set or retrieve a value <table border="1" style="margin-top: 10px;"> <tr> <td>0</td><td>Retrieves value</td></tr> <tr> <td>1</td><td>Sets value</td></tr> </table>	0	Retrieves value	1	Sets value
0	Retrieves value					
1	Sets value					
pValue		Pointer to an integer where the desired value is/should be stored				
Return parameters	Type	Description				
retval		0 Function successful				

See also: [VNC_SVR_NEVER_SHARED](#)

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCSVRNeverShared
VAR_INPUT
    set          : DINT;
    pValue       : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.24 VNCSVRReset



For internal use only. Do not use this function!

1.8.3.25 VNCSVRSetsDisplayOffset

For internal use only. Do not use this function!



1.8.3.26 VNCSVRSetsFilter

Which hosts are allowed to connect to the VNC server can be specified. Single rules can be combined to form a more complex filter by separating them with commas. It is important to keep in mind the first matching rule for a host is used.

See also: [VNC_SVR_FILTER](#)

Transfer parameters	Type	Description	
strFilter	^CHAR	This parameter defines the filter rules; the individual rules are separated by commas.	
		Rule	Example
		+<host>	Allows a specified host to connect.
			+192.168.1 0.30
		+<net>/<net mask>	Allows client from the specified subnet to connect to the server.
			+192.168.2 0.0 /255.255.25 5.0
		-<host>	Connection with the specified host is not allowed.
		-<net>/<net mask>	Does not allow clients from the specified subnet to connect with the server.
			- 192.168.50. 0 /255.255.25 5.0
Return parameters	Type	Description	
retVal	DINT	0	Function successful
		#0	Error

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCVRSetFilter
VAR_INPUT
    strFilter      : ^CHAR;
END_VAR
VAR_OUTPUT
    retval        : DINT;
END_VAR;
```

Example

The filter string -192.168.30.10,+192.168.30.0/255.255.255.0,- will allow any host from the subnet 192.168.30.xxx, except the single host with IP 192.168.30.10 (as the rule for that host appears before the subnet-rule). The last rule (default rule) '-' is to disallow any other host.

Another possibility could appear as follows:

The filter string -192.168.30.10,+ will allow any host except the one with the IP 192.168.30.10.



The default filter is '+', which means that all hosts are allowed to connect to the server!

1.8.3.27 VNCVRSetPass

This function is used to set a new password for the VNC-Server. To set the new a new password, the old one must first be entered for authentication.

Transfer parameters		Type	Description
oldPassword		^CHAR	String that contains the old password
Return parameters		Type	Description
retVal		DINT	VNC_OK OK, the new password has been set VNC_OLDPWD_IN Authentication failure – old password is VALID invalid

VNC_INVALID_PW DFILE	Password file has an invalid format
VNC_FAILED_OPE N_PWDFILE	Couldn't open password file
VNC_FAILED_WRIT ING_PWDFILE	Failed writing into password file

Prototype

```
FUNCTION __cdecl GLOBAL G_VNCsvrSetPass
    OldPassword      : ^CHAR;
    NewPassword      : ^CHAR;
VAR_OUTPUT
    retval          : DINT;
END_VAR;
```

The VNC server password can also be set using the following CLI command:

```
vncsvr setpass <oldpwd> <newpwd>
```



1.8.3.28 VNCsvrTimeoutInactiveClient

With this function, a time interval at which the VNC server disconnects an inactive client can be specified. The default value is 3600 seconds (The minimum value is 15 seconds). The default value is 3600 seconds (The minimum value is 15 seconds).

Transfer parameters	Type	Description	
set	DINT	Used to set or retrieve a value	
		0	Retrieve value
		1	Set the value
pSeconds	^DINT	Pointer to an integer where the desired value is/should be stored	
Return parameters	Type	Description	
RetVal	DINT	0 Function successful	

See also: [VNC_SVR_TIMEOUT_INACTIVE](#)

Prototype

```
FUNCTION __CDECL GLOBAL G_VNCVRTimeoutInactiveClient
VAR_INPUT
    set          : DINT;
    pSeconds    : ^DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

1.8.3.29 VNCVRUpdateRect

For internal use only. Do not use this function!



1.8.3.30 Error Codes VNC Server

Functions whose return parameter is a DINT have failed, if they return a value greater than 0.

Symbol name	Value	Description
VNC_OK	16#00	Function successful. No errors.
VNC_NO_SUCH_CLIENT	16#50	The client does not exist.
VNC_NO_SCREENUPDATE_MANAGER	16#51	The VNC server is not running.
VNC_INVALID_PWD_FILE	16#52	Password file is invalid.
VNC_OLDPWD_INVALID	16#53	Old password is invalid.
VNC_FAILED_OPEN_PWDFILE	16#54	Could not open password file for reading.
VNC_FAILED_WRITE_PWDFILE	16#55	Could not open password file for writing.
VNC_NO_SCREENBUFFER	16#56	Not enough memory to allocate pixel buffer.

1.8.4 Command Line Interface Commands

The VNC client service can be started or stopped manually with the vncsvr CLI command:

Syntax

```
vncsvr start | stop | setpass | version
```

Examples

```
vncsvr start [watch]
```

Will start the VNC server service.

Optionally “watch” can be entered to check in a 20 seconds interval, whether the VNC server still can accept incoming connections.

```
vncsvr stop
```

Will stop the VNC server service (all clients will be disconnected!)

```
vncsvr version
```

Shows the DLL version and the VNC protocol-version numbers of the VNC server.

```
vncsvr setpass <oldpass> <newpass>
```

Use this command to set a new VNC server password.

1.8.5 Configure the VNC Server Using Environment Variables

The VNC server can be configured using environment variables. They can be set them manually by entering each command in the CLI just before starting the VNC or, more comfortably, by defining them in the autoexec.lsl.

1.8.5.1 VNC_SVR_ACCEPT_KEYBOARD

This variable is used to set the server to accept remote keyboard events; if set to TRUE, the server accepts remote pointer events. When FALSE, remote pointer events are rejected.

Default value: TRUE

See also: [VNCsvrAcceptKeyboardEvents\(\)](#)

Example

```
SETENV VNC_SVR_ACCEPT_KEYBOARD TRUE
```

1.8.5.2 VNC_SVR_ACCEPT_POINTER

This variable is used to set the server to accept remote pointer events (mouse & touch); if set to TRUE, the server accepts remote pointer events. When FALSE, remote pointer events are rejected.

Default value: TRUE

See also: [VNCsvrAcceptPointerEvents\(\)](#)

Example

```
SETENV VNC_SVR_ACCEPT_POINTER TRUE
```

1.8.5.3 VNC_SVR_ALWAYS_SHARED

If this option is set to TRUE, the server treats all connection requests as if 'shared connection' option is set, regardless of what client specifies.

Default value: FALSE

See also: [VNCsvrAlwaysShared\(\)](#)

Example

```
SETENV VNC_SVR_NEVER_SHARED TRUE
```

1.8.5.4 VNC_SVR_BLACKLIST_LEVEL

Defines the point at which a defective VNC client is set to the internal blacklist.

- | | |
|---|--------------------------------------|
| 0 | Never, blacklist is deactivated |
| 1 | Test during authentication (default) |
| 2 | Test during connection setup |

Example

```
SETENV VNC_SVR_BLACKLIST_LEVEL 1
```

1.8.5.5 VNC_SVR_COMPARE_FB

This property defines, whether the server performs a pixel comparison on the frame buffer to reduce unnecessary updates (TRUE) or not (FALSE).

Default value: FALSE

See also: [VNCSVRCCompareFB](#)

Example

```
SETENV VNC_SVR_COMPARE_FB FALSE
```

1.8.5.6 VNC_SVR_DESKTOP_NAME

The desktop name of the VNC server can be defined with this environment variable. It is displayed in the title bar of the window of VNC clients. The maximum length of the name is 64 characters and spaces are allowed.

Default value: Sigmatek GmbH

Example

```
SETENV VNC_SVR_DESKTOP_NAME Name of the PLC
```

Requirements

The environment variable is supported from version 01.01.030 of the VNC Server DLM.

1.8.5.7 VNC_SVR_DISCONNECT_ON_NONSHARED

This variable defines the behavior of the VNC server when a client is connected. A VNC client specifies, whether share a server connection with other clients (shared connection), or not (non shared connection). The connection is allowed or refused depending on how the client is defined.

Default value: TRUE

See also: [VNCSVRCDisconnectClientsOnNonsharedConnection\(\)](#)

Example

```
SETENV VNC_SVR_DISCONNECT_ON_NONSHARED TRUE
```

1.8.5.8 VNC_SVR_DISABLE_SHARED_MEMORY

This environmental variable can prevent the Full Screen update from the shared memory, if more than 75% of the screen content has changed.

Default value: FALSE

Example

```
SETENV VNC_SVR_DISABLE_SHARED_MEMORY TRUE
```

Requirements

This feature is supported from interface version 01.01.037.

This feature is only supported in Salamander.

1.8.5.9 VNC_SVR_FILTER

This environment variable is used to set the filter string. For a detailed description, see [VNCVRSetFilter\(\)](#).

Default value: +

Example

```
SETENV VNC_SVR_FILTER +192.168.10.30,-
```

1.8.5.10 VNC_SVR_FORCE_BPP

Determines which color depth is propagated from the VNC server to the VNC client.

- | | |
|---|------------------------------|
| 0 | 16-bit color depth (default) |
| 1 | 8-bit color depth |
| 2 | 16-bit color depth |

Example

```
SETENV VNC_SVR_FORCE_BPP 1
```

1.8.5.11 VNC_SVR_FULL_UPDATE

This option allows you to force the VNC server to send a full screen update after the defined time interval. The time interval is specified in seconds.

Default value: 2

Example

```
SETENV VNC_SVR_FULL_UPDATE 4
```

1.8.5.12 VNC_SVR_LOGFILE

If an extra log file for the VNC server messages is desired, the following is necessary:

- The logging mechanism must first be told to write an extra log file by specifying the log string using [VNC_SVR_LOGSTR](#) to *:file:30 (an appropriate debug lever must be chosen).
- The file name must then be specified using VNC_SVR_LOGFILE.

Default value: C:\vncsvr.log

Example

```
SETENV VNC_SVR_LOGFILE c:\vncsvr.log
```

1.8.5.13 VNC_SVR_LOGSTR

This environment variable is used to define the log string. For a detailed description, see Log string description.

Default value: *:kernel_svr:0

Example

```
SETENV VNC_SVR_LOGSTR *:kernel_svr:30
```

1.8.5.14 VNC_SVR_MAX_CONNECTIONS

Defines how many simultaneously connections the server allows.

Default value: 2 with systems equal to or lower than 256 MB RAM, 4 with systems higher than 256 MB RAM

With Salamander the number is limited to 6.



See also: [VNC_SVRMaxConnections](#)

Example

```
SETENV VNC_SVR_MAX_CONNECTIONS 2
```

1.8.5.15 VNC_SVR_NEVER_SHARED

If this option is set to TRUE, the server does not share a connection, regardless of what is specified by the client.

Default value: FALSE

See also: [VNC_SVRNeverShared\(\)](#)

Example

```
SETENV VNC_SVR_NEVER_SHARED TRUE
```

1.8.5.16 VNC_SVR_PORT

Defines the port number that the VNC server should use.

Default value: 5900

Example

```
SETENV VNC_SVR_PORT 5900
```

1.8.5.17 VNC_SVR_PORT2

Defines a second port number being used by the VNC server.

Default value: As a default the VNC server only uses the port number given in [VNC_SVR_PORT](#).

Example

```
SETENV VNC_SVR_PORT2 6900
```

1.8.5.18 VNC_SVR_TIMEOUT_INACTIVE

Defines the time in seconds after which the VNC server disconnects an idle client.

Default value: 3600

Example

```
SETENV VNC_SVR_TIMEOUT_INACTIVE 3600
```

1.8.5.19 VNC_SVR_WAIT_FOR_CLIENT

This property defines how long (in milliseconds) the server waits for a client, which does not respond.

Default value: 20000

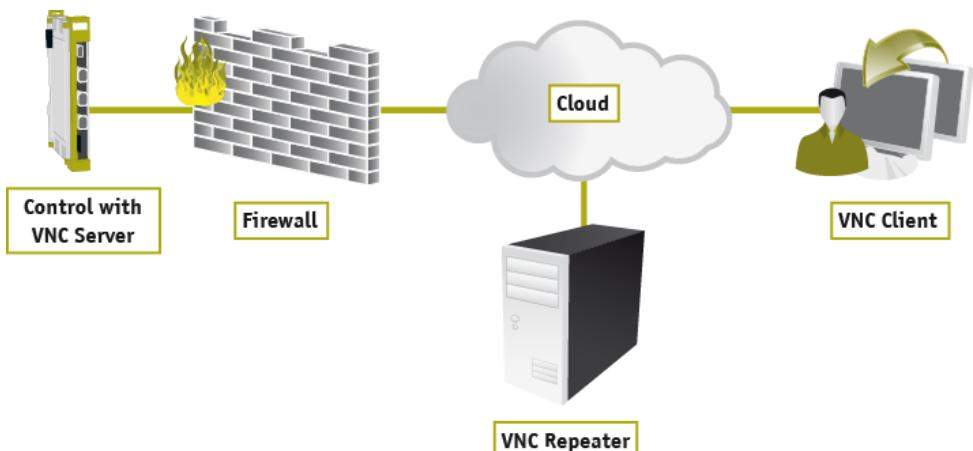
See also: [VNCsvrClientWaitTimeMillis\(\)](#)

Example

```
SETENV VNC_SVR_WAIT_FOR_CLIENT 50000
```

1.8.6 Repeater Function

The VNC server DLM for SIGMATEK controls have a repeater extension. This is used to create the connection to VNC servers (controls) that are not accessible in the network (located behind a firewall for example). Controls with the VNC server and VNC clients from a connection to a repeater which performs data exchange.



So that the repeater is started, a valid vncsvr.cfg must be available.
The extension is available starting from version 1.1.020.

1.8.6.1 Starting the Repeater Function

The DLM is started, as usual, with vncsvr start in the CLI or Autoexec.lsl. The repeater function is started when a valid vncsvr.cfg is available; no further inputs are required.

Before use, a password must first be entered.

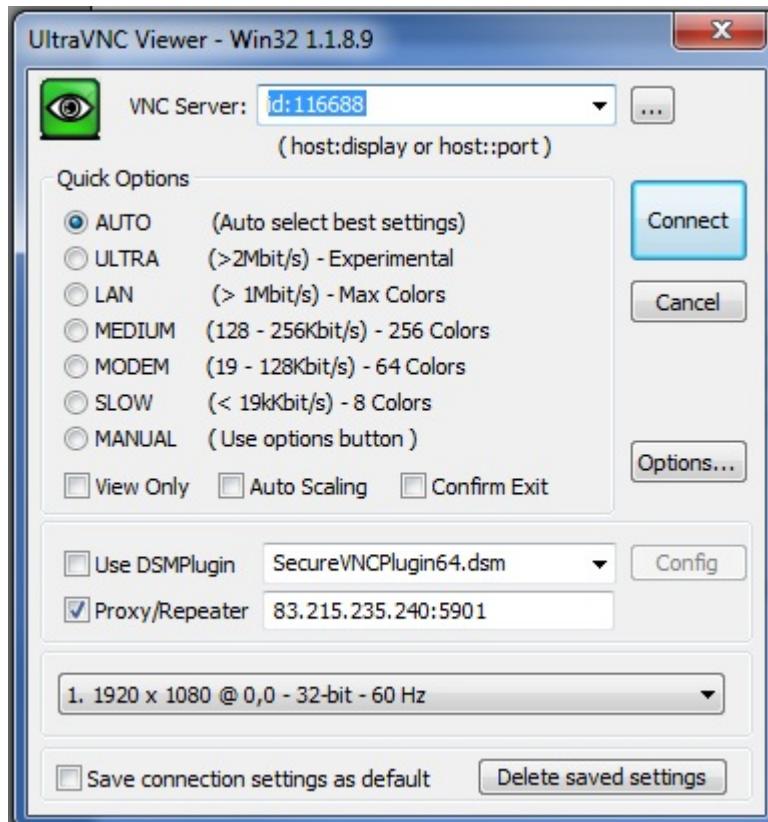
```
vncsvr setpass Sigmatek mypass
```

Under some conditions, the creation of a gateway is required (consult the network administrator for this information).

```
set ip gateway 192.168.1.1
```

1.8.6.2 Using the Repeater Function

Any VNC client can be used as a client in which a repeater (proxy) is entered. Here, Ultra VNC is used as an example.



The ID from vncsvr.cfg is used as the VNC-Server. Here, "id" must be written before the actual ID. IP from vncsvr.cfg and the port (5901) are entered under Proxy/Repeater. The password is then requested before the connection is established.

1.8.6.3 Configuring the Repeater Function

The vncsvr.cfg configuration file, which is a text file that must be stored in the C:\LSLSYS folder, starts the repeater function. In order to be valid, this file must contain the following key value pairs.

```
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5500
REPEATER_ID = $SERIALNUM
```

The REPEATER-IP contains the IP and REPEATER-PORT, the port of the repeater that should be used. DNS can also be defined as IP.

No individual ID can be assigned under REPEATER ID. This ID must be numeric and can be a maximum of eight characters. If \$SERIALNUM is entered as the value, the VNC server logs into the repeater with the serial number of the control. Commentary can be set using a hash "#".

1.8.7 Sample Configurations Using the AUTOEXEC.LSL

1.8.7.1 Scenario 1: The server should allow exactly one connection from a specific host.

In the first scenario the VNC server should be configured to accept only one connection from a specific IP address (e.g. 192.168.30.45) and accept remote keyboard and touch/mouse input. The VNC service should scan port 6300 and disconnect idle clients after 2 hours (7200 seconds).

- First ensure that the VNC server is installed correctly.
- Edit the AUTOEXEC.LSL and add the following rules:

```
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 Scenario 2: The server should allow every host to connect, but only for viewing.

In the second scenario the VNC server should be configured to accept every incoming connection request up to 5 connected clients but no remote input. The VNC service should scan the default port 5900 and disconnect idle clients after 8 hours (28800 seconds).

- First ensure that the VNC server is installed correctly.
- Edit the AUTOEXEC.LSL an add the following rules:

```
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 #####
VNCUSR START
```

1.8.7.3 Scenario 3: Setting up the server using a separate log file.

In the second third the VNC server should be configured to accept every incoming connection request from all IPs from a specific subnet (192.168.30.0) up to 4 connected clients and remote input. The VNC server should scan the default port 5900 and write its log data to the c:\vncsvr.log file, to obtain a detailed log file (including debug messages).

- First ensure that the VNC server is installed correctly.
- Edit the AUTOEXEC.LSL an add the following rules:

```
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 #####
VNCUSR START
```

1.8.7.4 Scenario 4: Set up a server which will write remote events into the user-log event02.log.

In the fourth scenario all remote commands (Keyboard and Mouse/Touch events) from the VNC clients should be logged in the event02.log user log file. The following lines must therefore be added to the autoexec.lsl:

```
REM ##### TURN ON EVENT - LOGGING #####
SET USERLOG2 ON
REM #####
```

1.8.8 Example Configurations with the vncsvr.cfg File

The vncsvr.cfg file is required to access the the VNC server through a repeater (see [Repeater](#)).

1.8.8.1 Scenario 1: VNC server logs in with the serial number

In the first scenario, the VNC server logs into the repeater with the serial number. The following lines must therefore be added to the vncsvr.cfg file:

```
##### Configuration - Repeater Logged-On with Serial number#####
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5500
REPEATER_ID = $SERIALNUM
```

1.8.8.2 Scenario 2: VNC server logs in with an individual ID

In the second scenario, the VNC server logs into the repeater with an individual ID (113355). The following lines must therefore be added to the vncsvr.cfg file:

```
##### Configuration - Repeater Logged-On with individual ID #####
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5500
REPEATER_ID = 113355
```

1.8.8.3 Scenario 3: VNC-Server logs in with DNS

In the third scenario, the VNC server logs into the repeater with a name (DNS). The following lines must therefore be added to the vncsvr.cfg file:

```
##### Configuration - Repeater Logged-On with DNS #####
REPEATER_IP = Sigmatek.at/repeater
REPEATER_PORT = 5500
REPEATER_ID = $SERIALNUM
```

1.9 SRAM

1.9.1 Formats

With the SRAMFORMAT (SETENV SRAMFORMAT 1|2) environment variable, which SRAM format to use is set.

The SRAMFORMAT has nothing to do with the SET RAM FORMAT NEWW setting. This setting is obsolete and does not have to be used. In the C-IPC, TEACHBOX and EDGE2 platforms format 2 is used by default, in the remaining platforms format 1 is used.

When an SRAM format is set with SETENV SRAMFORMAT, this command should also be contained in the Autoexec.lsl.

1.9.1.1 Format 1

Data and control structures of the RAM and RAMEx objects are found in the SRAM.

Files:

C:\LSLDATA\SRAM.CPY ... temporary backup for the SRAM-Reorg is used during start-up.

As is this format the control structures lie in the Sram and the Sram memory has a certain fragmentation, the value shown in the LASAL Class in the Project Info window is only an approximate value, i.e. from this value you cannot precisely determine, how many RAM or RAMEX objects can still be created.

1.9.1.2 Format 2

This format can only be used on platforms with file system.

Static control structures of the RAM and RAMEx objects are located in a file, the data is in SRAM. The static control structures change the start of the application when RAM/RAMEx objects are added or removed.

The advantage of these formats is that the entire SRAM is available for data and more data can therefore be stored.

Files:

C:\RAMFILE.DAT ... static control structures

C:\LSLDATA\RAMFILE.CPY

C:\LSLDATA\SRAM.CPY ... temporary backup for the SRAM-Reorg is used during start-up.

1.9.1.3 Converting between Formats

Starting from loader version 02.02.123, SRAM data can be converted between the formats.

The data is automatically converted when the application starts. The format of the actual SRAM content is compared with the setting in the SRAMFORMAT environment variable. If the formats are different, the data is converted.

It is important to note that conversion from format 2 to format 1 is then only possible if there is enough space in the SRAM for data in format 1, as in this format SRAM data uses significantly more memory.

If conversion is not possible, the data remains in format 2.

1.9.2 Retentive Servers

A retentive server can retain a 4-byte value.

If set in an SRAM server, the data is saved in SRAM. With the FILE setting, the data is stored in the "C:\RETSVR.DAT" file.

If the setting changed from SRAM to FILE or vice versa, then with version 02.02.123 or higher, the data is assumed by the other medium.

Starting from version 02.02.123, the LDR_IsAsncFileOperationInProgress function can be used to determine whether asynchronous file operations are still active (Retentive File). This function is required when writing to several retentive servers (e.g. when loading a tool catalog) and a reboot is then performed.

Because the data is stored in the backup file when writing to a retentive server asynchronously to the application program, rebooting is only possible when the data in the retentive server is backed up.

1.9.3 RamEx

Starting from RamEx version 1.6, MerkerEx version 1.2, StringInternal version 1.5 and loader version 02.02.123, RamEx data can be stored in respective file. The UseFile client must thereby be set to 1 - this only possible with InitValue in LASAL.

The files are stored in the "C:\LSLDATA\RAMEX" directory. Additionally, a RAMEX.IDX index file is stored in the C:\LSLDATA directory, in which the valid RamEx files are entered.

If a RamEx object is later configured so that the data is stored in a file and SRAM data is already available, the data from the SRAM is assumed and stored in the file. The SRAM entry is deleted during reorganization.

The data is also taken over when a RamEx object is reconfigured from File to SRAM. The file is deleted.

1.9.4 Save, Restore, Delete

1.9.4.1 Backing up the SRAM

With the CLI command [SRAMSAVE](#), SRAM data can be backed up in a file.

SRAMSAVE <filename>

Backs up the SRAM content in a file.

SRAMLOAD <filename>

Restores the SRAM content from a backup file.

Starting from OS version 01.02.195, retentive file data and RamEx files are also backed up. It is still possible to load old backup files (without RETSVR.Dat and RamEx files).

With backing up and restoring data using the [SRAMSAVE](#) CLI command, the format cannot be converted. This means that it is not possible to back up an SRAM saved in format 2 with format 1.

1.9.4.2 Deleting the SRAM

SRAMCLEAR

Deletes remnant data stored in the SRAM (RAM, RAMEx, retentive SRAM server).

SRAMCLEARFILE

Deletes remnant data backed up in the file (File server).

SRAMCLEARALL

Deletes all remnant data (in SRAM as well as in the file).

1.9.4.3 Loading a Reorganisation Copy

Starting from loader version 02.02.123, an SRAM copy created during reorganization of the SRAM can be reloaded.

The *.CPY files in the C:\LSLDATA directory must be renamed to *.INI. If SRAMFORMAT 1 is set, the SRAM.INI file is then required - with format 2, the "SRAM.INI" and "RAMFILE.INI" files are needed.

The SRAM format must match the INI files. After loading the files, they are renamed *.DON (DONE).

These files are loaded while reorganizing (after reboot) the SRAM when available in the LSLDATA directory and the SRAM is valid (not CPU status "Ldr out of near" or from OS-version 01.03.090 „SRAM-Error“).

With a new CPU (the "isnotnew.inf" file is NOT available) the files are also loaded when the SRAM is invalid (with first RUN after PowerON only).

If, for example, the CF card a C-IPC is inserted in a new one and the SRAM data (version from the last reboot) should be loaded, the *CPY files in the LSADATA directory must be renamed to *.INI and the isnotnew.inf file must be deleted.

1.9.4.4 LASAL CLASS Tool Ram Image

Also with the LASAL CLASS tool RAM Image SRAM data can be saved, restored and deleted. The functionality of this tool is described in the LASAL online help.

1.9.5 SRAMDisk

With the Edge platform, the null-voltage safe data is stored in RAM and (or only) are saved to a reserved space on the SD card (=SRAMDisk) in the background cyclically during shutdown.

The behavior of the SRAM data backup can be set with the following parameters. The values can be queried or changed using the CLI command SET SRAMDISK or the LASAL CLI interface "SRAMDISK".

Parameter name	Definition
NPD (Default 32)	N-PowerDownSectors. The number of sectors to which can be written during power down. The size of a sector is 512 bytes.
T1 (Default 3)	T1 determines how quickly the number of changed sectors should be set to a value < NPD within a certain time. (Unit: seconds, -1 = value is inactive)
T2 (Default 30)	T2 determines the time elapsed after which the changed sectors are written to the disk, also when the number of changed sectors is at a value < NPD. (Unit: seconds, -1 = value is inactive)
T3 (Default 60)	T3 determines how often consistent version is created on the disk and changed to a new version number within a certain time. (Unit: seconds, -1 = value is inactive)
FBFW (Default 1)	Flush Before File Write. If FBFW is at value ≠ 0, all changed SRAM data is written to the disk before writing to a file.
ONLYPD (Default 0)	If ONLYPD is at a value ≠ 0, no SRAM data is written to the disk during the running operation (during PowerDown only). In difference to the parameter SramdiskFullCopyAtPowerdown (see description below), here SRAM writing operations are detected by the MMU further on, which increases the CPU load. So if cyclic backup of the SRAM data in the background is renounced, the parameter SramdiskFullCopyAtPowerdown has to be preferred.

In the optional C:\LSLSYS\CONFIG.LSL file, the settings for the SRAMDisk size and power down response are found.

Parameter name	Definition
SramSize (Default 524288)	SRAM size. A value divisible by 4096 must be set.
SramdiskFullCopyAtPowerdown (Default 0)	With a value of 1, the entire SRAM is written to the disk regardless of whether the SRAM content has changed or not. No SRAM data is written to the disk in the background.

Starting from OS version 01.02.195.

Parameter name	Definition
SysSramSize	Size of the SRAM area used for the operating system (e.g. for the temporary buffer with the Event log file). A minimum size of 8 k is recommended. If the value is too small, a log message in the Event log file could be lost. It is important to note that changing this parameter also changes the SRAM size for the application. This means that when sysSramSize is increased, the application has less SRAM available, which can result in SRAM data loss. If this parameter is not configured, the size is calculated as follows.

```
if sramSize ≥ 128 k then sysSramSize = 16 k
else if sramSize ≥ 32 k then sysSramSize = sramSize / 8
else sysSramSize = sramSize / 16
```

With "scramdiskFullCopyAtPowerdown = 1" and a sramSize value of 16 kBytes, the writing rate on the SD card can be reduced to a minimum and the lifespan of the card can be thereby extended. From these 16 kBytes the operating system needs 4 kBytes to safely write event log files. 12 kbytes therefore remain for the user; this corresponds to approximately 3000 SRAM values.

Starting from OS version 01.02.195, the data is retained when the size is changed as long as long as there is enough space with a reduction of the disk.

Starting from OS version 01.02.195, data is also written to the SRAMDisk if the SRAM is configured so that the data is written during PowerDown only (ONLYPD or SramdiskFullCopyAtPowerdown).

1.9.5.1 SRAMDisk Backup Mechanisms

It is distinguished between backup during running operation and backup only when switching off.

In both cases there are two areas on the SD card, where one of these areas always contains a completed consistent image of the SRAM and the new data to be backed up are written to the other area.

1.9.5.1.1 Backing Up Data during Operation and when Switching Off

Remanent data are in the RAM and after a change during operation they are written to the SD card in the background. When switching off, the rest of the data not written so far are written to the SD card.

The remanent data in the RAM are write protected pagewise by the MMU (Memory Management Unit of the CPU). As soon as the user writes to an area, an exception in the operating system occurs. The size of a page is 4k, in the exception the address written to is available, so the operating system knows, when data in the remanent area changed. Then the write protection of the page is removed.

A 4k page is further split into 8x 512 byte sectors, which is the minimum write size of a SD card.

Cyclically the sectors of an unlocked page are checked, whether the contents changed compared to a backup page mirroring the contents of the SD card. If this is the case, this sector is marked as changed. Changed sectors then are written to the SD card in the background. As soon as all sectors of a page have been written to the SD card, the write protection of the page is activated again. The frequency of writing changed sectors to the SD card is controlled by the parameters NPD, T1, T2 and T3.

So changes of remanent data cause writing operations to the SD card, which shortens the lifetime of the SD card. The more and the more frequently remanent data are changed, the more writing operations on the SD card occur.

Also the distribution of data in the SRAM plays a role here. The closer the data are next to each other, the fewer sectors are affected by changes and the fewer SD card writing operations are needed. But the user cannot influence the distribution of data.

1.9.5.1.2 Backing Up Data Only When Switching Off

Remanent data are in the RAM and only are written to the SD card when switching off.

So changes of remanent data do not cause writing operations to the SD card during runtime and no higher CPU load due to background processing.

So in this setting the size of remanent data can only be so high, that the time for backing up when switching off is long enough. The value here is 16 kbytes. From these 16 kbytes the operating system needs 4 kbytes to safely write event log files. 12 kbytes therefore remain for the user; this corresponds to approximately 3000 SRAM values.

1.9.5.1.3 Advantages and Disadvantages of these Two Ways of Backing Up

- Data loss if writing to the SD card does not work when switching off
 - If data are backed up during operation, a set of data at the maximum some minutes old is still available. If data are saved only when switching off, the last available data is the one of switching on.
- Stress of the SD card due to writing operations
 - When backing up data during operation the number of writing operations on the SD card is higher. The number depends on the number of SRAM changes, the distribution of SRAM data and the used settings.
- CPU load
 - When backing up data during operation, the CPU load is higher due to the backing up in the background and the unlocking of a page for writing to the SRAM.

1.9.5.1.4 Monitoring the Number of Writing Operations to the SD Card

The number of sector writing operations to the SD card are counted and logged by the operating system. The current counter readings are also accessible via system variables.

The counter readings are written to the file `c:\sysmsg\<volume-serial>.wst` every 10 minutes, where `\<volume-serial>` is the serial number of the partition of the SD card. The contents of the file has the following structure:

```
<N1> writes, <N2> seconds, <N3> sramdisk-writes
```

N1	Total number of the sector writing operations
N2	Time how long the SD card is already in use (in seconds)
N3	Number of sector writing operations due to SRAM disk backups

The following system variables allow to access these values:

<code>_diskOperatingSeconds</code>	Time how long the SD card is already in use (in seconds)
<code>_diskSectWrCnt</code>	Total number of the sector writing operations
<code>_diskSectWrCnt1000Sec</code>	Number of sector writing operations in the last 1000 seconds
<code>_diskSectWrCnt10Sec</code>	Number of sector writing operations in the last 10 seconds

_diskSectWrCn	Number of sector writing operations due to SRAM disk backups
tSramdisk	

1.9.5.2 Detecting an Error when Writing to the SRAMDisk during Power-down

Starting from OS version 01.03.011, information as to whether or not it is possible to write to the SRAMDisk during power-down is provided. Starting from OS version 01.03.070, this information is only provided, if cyclic backup (sramdiskFullCopyAtPowerdown = 1) is not activated.

This information is evaluated by the loader starting with version 02.02.158 and in the event of an error, an SRAM error is triggered in the loader (the SRAM application data is thereby reset).

This response from the loader can be deactivated starting with loader version 2.2.167:

If the environment variable NO_SRAM_POWERDOWN_CHECK exists and has a value unlike "0", the test to check if the SRAM was written during power-down is then skipped.

Example

```
SETENV NO_SRAM_POWERDOWN_CHECK 1
```

1.9.6 Response when an Error is Detected in the SRAM

If the loader detects an error in the SRAM data structure, the entire SRAM is set to 0 or to the initial value and the application starts in halted (CPU status Ldr.Out-Of-Near or from OS-version 01.03.090 „SRAM-Error“). The control must then be manually restarted. This ensures that the control does not start with the initialization values without being noticed.

If in the event of an error, only the SRAM is set to the Init values and the application should start, the environment variable SRAM_CONTINUE_ON_ERROR must be set to a value unlike 0.

```
SETENV SRAM_CONTINUE_ON_ERROR 1
```

1.9.7 Reorganizing the SRAM

With the first application start after power up or changing projects, the SRAM on platforms with file system is reorganized. The SRAM content is thereby written to a temporary copy, the SRAM is then reconstructed. If the value is available in the copy, it is used. Otherwise the Init value set in the project is used. Through the reorganization the unnecessary Remix objects are removed from the SRAM.

The unneeded RamEx files are deleted through the reorganization.

If reorganization should be performed after power up as well as after a later "RESET+RUN", the LDR_ForceSramReorgOnNextRun function must be called.

Calling this function results in the reorganization of the Sram with the next RUN.

If the environment variable SRAM_DISABLE_REORG is set to a value unlike 0, the reorganization of the SRAM is suppressed

Files

C:\LSLDATA\REORGINF.DAT

In this file, the project name and an OS instance ID is stored. With these values, the loader can determine whether the reorganization is being performed with the first application start after power up or a change to a different project has been made.

1.9.8 SRAM Size and Memory Allocation

The size of the physically available SRAM can be found in the CPU datasheet:

Example, CP731-K CP 731-K.pdf

Internal data memory (SRAM) 512-kbyte (battery buffered)

A part of this memory space is used for the operating system. In Salamander CPUs, 16 kB are normally used. The rest is provided for the application for zero-voltage protected objects (RAM and RAMEx objects, retentive server Sram).

The concrete value can be queried via the _MemoryInformation class or the global variable _S_Ram_Hptr^.DataLength.

Example, CP 731-K

_MemoryInformation.SRAMTotal 507 840

_S_Ram_Hptr^.DataLength 507 840

The number of RAM and RAMEx objects that can then actually be used depends on the Sram format used. With Sram Format 1, static management information is also stored in the Sram. That was changed with Sram Format 2. Here, a large part of the static management information is stored in the C:\ramfile.dat file. Sram format 1 is obsolete and no longer used in Salamander CPUs. Therefore, it will not be discussed here in detail.

Memory allocation in Sram Format 2:

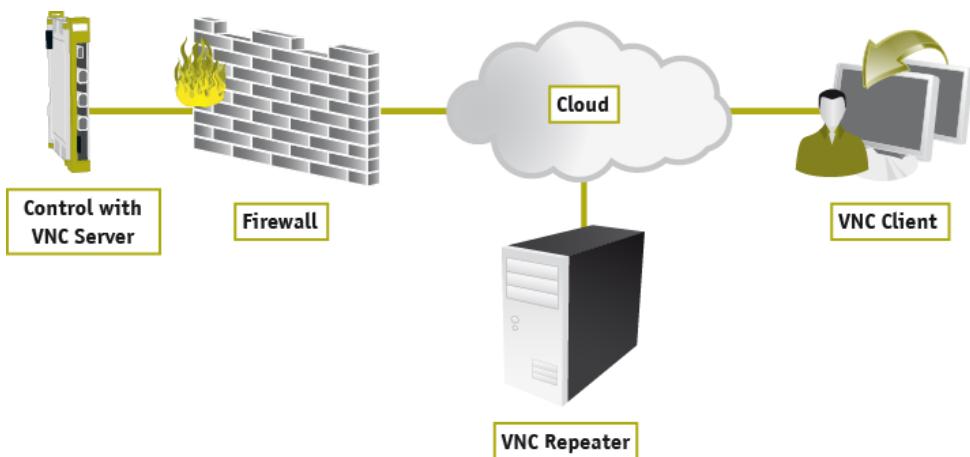
- The Sram memory, which the application provides, is allocated as follows:
- In any case a 72-byte header is required
- 4 bytes are required per RAM object

- A retentive server Sram has the same memory requirements as a RAM object
- Per RAMEx object, a 16-byte header plus the actual RAMEx data is required.

The memory required for the RAMEx data is not allocated bytewise, but in 4-bytes units. Frequently changing the size of the RAMEx data can fragment the memory, resulting in unused Sram memory. Example: If 2 RAMEx objects exist, each with 1000 bytes, and the size of the first RAMEx object is reduced to 800 bytes, 200 bytes of unused SRAM space is generated

1.10 Repeater

The LASAL Repeater function for SIGMATEK controls was designed to connect with controls that cannot accept (allow) any connections from the Internet. Controls with the LASAL Repeater function establish a connection to a repeater. The connection is maintained as long as the repeater function in the control is active. After the control has logged into the server, the standard ports on the control are provided to the repeater server.



1.10.1 LASAL OS with Repeater Function

1.10.1.1 Configuration

So that the LASAL Repeater can be started, a valid `lslrepeater.cfg` must be available under `C:\LSLSYS`. The login with the corresponding ID and password must also be stored in the repeater server, since it manages a white list containing all valid IDs. Only when this ID is entered in the server, can the control connect and a connection with LASALCLASS made.

1.10.1.2 `lslrepeater.cfg`

This file must contain the following key value pairs. The obligatory parameters must be available, so that the file is valid and the control establishes a connection to the server and logs in.

Comments can be set with a hash "#".

Obligatory Parameters

```
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5510
REPEATER_ID = $SERIALNUM
PASSWORD = Sigmatek
```

The REPEATER-IP contains the IP and REPEATER-PORT, the repeater port that should be used. As IP also a DNS can be defined.

Under REPEATER-ID, the individual ID can be assigned. This ID must be numeric with a maximum of 8 digits. If `$SERIALNUM` is entered as the value, the control logs into the repeater with the serial number.

PASSWORD is the password set in the server for the ID. This is requested while establishing the connection (stored in the repeater server). The password is not the online password entered when going online via LASALCLASS and set in the control through "set onlinepwd".

Optional Parameters

```
KEEPALIVE_INTERVAL = 30 #Default 20
KEEPALIVE_RETRY = 30 #Default 20
KEEPALIVE_TMO = 30 #Default 20
TCP_WINDOW_SIZE = 12288 #Default 32k
```

`KEEPALIVE_INTERVAL`: the time in seconds the must elapse before a Keep-Alive packet is sent to the repeater server.

KEEPALIVE_RETRY: the time in seconds between 2 Keep-Alive packets.

KEEPALIVE_TMO: if the counterpart is not reached, the connection is terminated after the specified time (in seconds).

TCP_WINDOW_SIZE: the size of the window in the TCP/IP protocol (can be between 4*1460 and 40 kbytes).

1.10.1.3 Command

```
lslrpt start
```

The function is started with lslrpt start CLI or in the Autoexec.lsl.

```
lslrpt stop
```

The function is deactivated with lslrpt start CLI or in the Autoexec.lsl.

Under some conditions, a gateway must also be set (please consult your network administrator for this information).

```
set ip gateway 192.168.1.1
```

1.10.2 Example Configuration with the Islrepeater.cfg File

The Islrepeater.cfg file is required to connect to the control via a repeater (see [Repeater](#)).

1.10.2.1 Scenario 1: Control Logs in with Serial Number

In the first scenario, the control logs in to the repeater with the serial number. The following lines must therefore be added to the Islrepeater.cfg file:

```
##### 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 Scenario 2: Control Logs in with Individual ID

In the second scenario the control logs in to the repeater with an individual ID (113355).

The following lines must therefore be added to the Islrepeater.cfg file.

```
##### Configuration - Repeater Logged-On with individual ID#####
REPEATER_IP = 83.215.235.240
REPEATER_PORT = 5510
REPEATER_ID = 113355
PASSWORD = Sigmatek
```

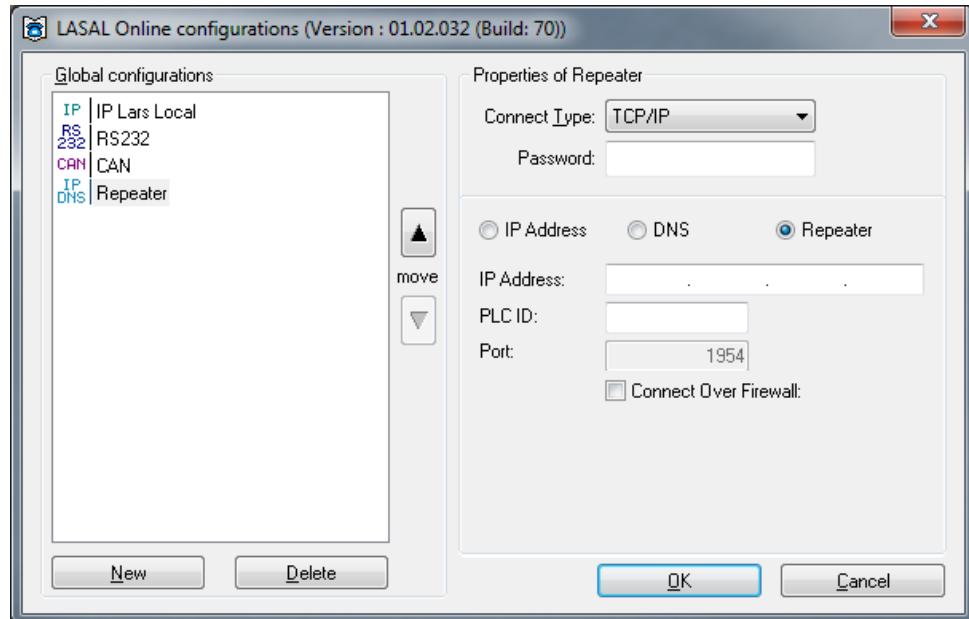
1.10.2.3 Scenario 3: Control Logs in via DNS

In the third scenario the control logs in to the repeater via a name (DNS). The following lines must therefore be added to the `lslrepeater.cfg` file.

```
##### 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

In the LASAL online configuration, the TCP/IP connection is expanded with the repeater. In IP Address, the repeater server IP is entered.



In the PLC ID field, the number under which the control logged into the repeater server (corresponds to the `REPEATER_ID` in the `lslrepeater.cfg` file).

The Password field can remain empty if no online password was assigned. It therefore depends on the password stored in the control with set `onlinepwd`. This password has nothing to do with the repeater and the password entered therein (this is defined in the `lslrepeater.cfg` file).

The Connect Over Firewall field has no direct involvement with the repeater configuration and can remain unchanged (it is possible that a firewall packet is not sent over the entered connection - see LASAL CLASS documentation).

1.10.3 LASAL Repeater

The server was integrated into the existing VNC repeater and the user interface was expanded. The IDs and the corresponding passwords of the controls must be stored in the repeater. For release of the ID's, please contact the provider of the respective server.

1.11 Remote Diagnosis

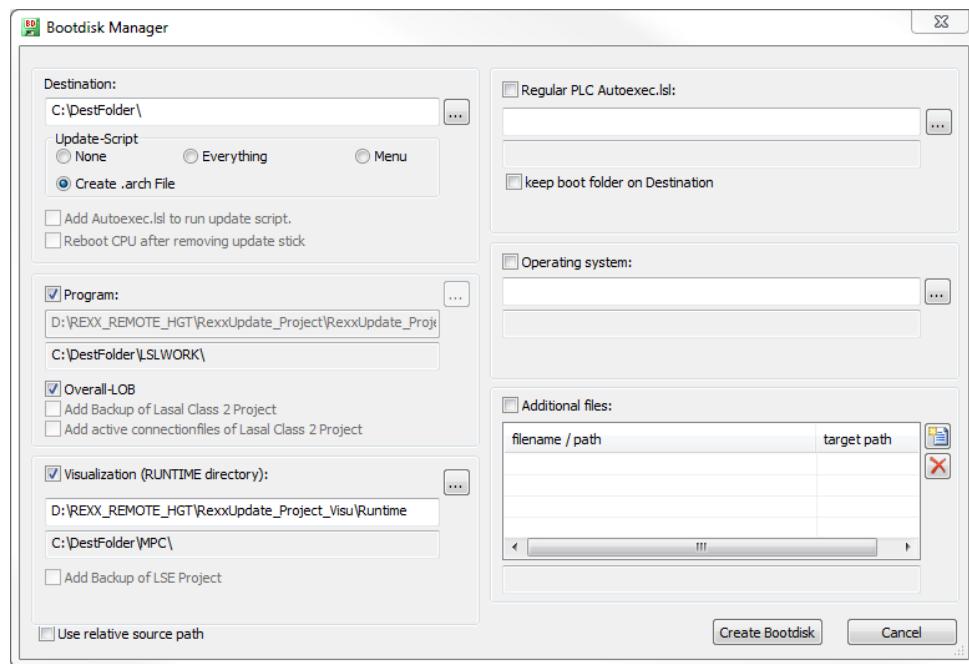
During remote diagnosis, the diagnostic control is connected via Ethernet with the control to test. A script is run after starting the diagnostic control, which automatically checks the test control for new a new application (tested by sending a checksum). If a new application is found (packed as an .arch file), the file is sent to the diagnostic control, extracted and started.

The "diagnostic application" must be created specifically for the project. It is important to note that the visualization is run as a multi-CPU solution so that the diagnostic control can display the values of the test control. For this purpose, the IP address of the test control must be entered in LSE under "Reference to Variables".

Possible error sources: The IP address is permanently stored in the LSE project, if changed by the customer, the IP could then be incorrect and therefore disable communication!

Installing the Controls

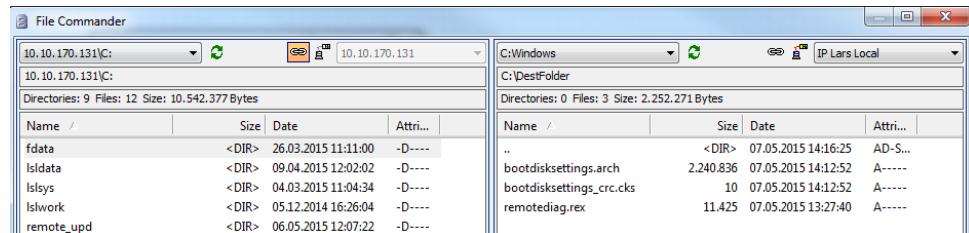
In LASAL CLASS 2, the CreateBootdisk function (menu bar Tools => CreateBootdisk) can be used to archive a diagnostic application (.arc file). At the same time, a checksum file (.cks file) and diagnostic script (.rex file) are generated. The generated files are stored in the folder defined in the bootdisk manager in the "Destination" field.



In the "Update Script" section, the selection field "Create .arch file" must be activated in order to create a compressed archive file from the project file (.lcp file) specified under "Program" and the (optional) selected visualization (Runtime folder). A checksum file and diagnostic script are also copied to the specified destination folder.

The files are created by pressing the "Create Bootdisk" button.

If the message "Create Bootdisk finished successful" appears, the generated file can be taken from the destination folder and copied to the corresponding controls.



1.) The "bootdisksettings.arch" and "bootdisksettings_crc.cks" files must be stored in the test control (e.g.: CP 111) in the main directory "C:\\" in the "remote_upd" folder (if this

folder does not exist, it can be created with the FileCommander function "MAKE DIR"). The memory location and folder name can also be change in the .rex file "remoteDiag.rex" if required (see Rexx Configurations).

2.) The "remotediag.rex" file must be copied to the diagnostic control (e.g.: HGT 835) in the main directory "C:\".



In the autoexec.lsl in the diagnostic control, the "QUCMD REXX remotediag.rex" call must be available so that the diagnostic process is run during start-up.

ED IT Edit File 'Autoexec.lsl' X

```

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
REM EVENTLOG ON
REM OSHEAP 100000
REM IP HOSTADDR 10 10 170 107
REM IP SUBNET 255 0 0 0
REM RUNTIME 30
REM CALIB CHECK
REM
REM QUCMD REXX remotediag.rex

```

Reload File Save on PC Save on PLC Save & Reboot PLC Cancel

3.) Rexx configurations: The section for user definitions are located at the start of the script (these definitions are based on the test control).

Here, the appropriate IP address of the test control must be set.

The path in which the .arch file and the checksum file are located can also be changed.

```
-----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 Operating System Interfaces

2.1 Variables

Name	Location	Data type	Description
_RealMaximumTime	(*AT % M 0000*)	: UDINT (4 bytes unsigned)	Maximum duration of the Real-time methods.
_RealAverageTime	(*AT % M 0004*)	: UDINT (4 bytes unsigned)	Average duration of the RT methods.
_CyclicMaximumTime	(*AT % M 0008*)	: UDINT (4 bytes unsigned)	Maximum duration of the Cy methods.
_CyclicAverageTime	(*AT % M 000C*)	: UDINT (4 bytes unsigned)	Average duration of the Cy methods.
_BackgroundMaximumTime	(*AT % M 0010*)	: UDINT (4 bytes unsigned)	Maximum duration of the Back-ground methods.
_BackgroundAverageTime	(*AT % M 0014*)	: UDINT (4 bytes unsigned)	Average duration of the Back-ground methods.

These values are always available for all PLCs up to LasalOS Version 01.01.050.

Starting from LasalOS Version 01.01.050 for small CPUs (CCL081, CCL721, CCL722, DCL642), these measures are disabled by default (due to performance reasons) and must be activated via the following system variable:

_SysTaskMeasEnable	(*AT % M 0319*)	: USINT (1 bytes unsigned)	Enables/Disables measure
--------------------	-----------------	----------------------------	--------------------------

If set to 0, the measure of Realtime, Cyclic and Background maximum and average time is disabled (gains performance on small CPUs); otherwise they are enabled.

_rebootOnErrorHandler	(*AT % M 0316*)	: USINT (1 bytes unsigned)	Enables reboot on error
-----------------------	-----------------	----------------------------	-------------------------

If the default value is 0, the CPU does not reboot when an error occurs. If set to a value \neq 0, the system reboots after a few seconds.

_isRebootCodeAvailable	(*AT % M 0317*)	: USINT (1 bytes unsigned)	
------------------------	-----------------	----------------------------	--

If this variable is "1", then variable "_rebootCode" is valid.

_rebootCode	(*AT % M 0317*)	: USINT (1 bytes unsigned)	Enables/Disables measure
-------------	-----------------	----------------------------	--------------------------

This variable contains the CPU-status before the reboot when an error occurs. With this value, the cause of the reboot can be determined.

_RTOSversion	(*AT % M 02EC*)	: UDINT (4 bytes unsigned)	Version of the PLC operating system.
--------------	-----------------	----------------------------	--------------------------------------

Setup of the _RtOSversion variable (4-byte):

3	2	1	0	nth byte of the variable
+	-	-	-	+
	0		5	11
+	-	-	-	+
				-----> Minor number of the OS
			+	-----> Major number of the OS
			+	-----> not used
			+	-----> not used
			+	-----> not used

_UserProgPointer	(*AT % M 02F0*)	: ^USINT (pointer on 4 bytes unsigned)	Address of the User CODES.
_UserProgSize	(*AT % M 02F4*)	: UDINT (4 bytes unsigned)	Size of the user-executable program (Code).
_UserDataPointer	(*AT % M 02F8*)	: ^USINT (pointer on 4 bytes unsigned)	Address of the user DATA memory.
_UserDataSize	(*AT % M 02FC*)	: UDINT (4 bytes unsigned)	Size of the user DATA memory.
_cpuLoad	(*AT % M 0EA6*)	: UINT (2 byte unsigned);	CPU load.

This variable contains the CPU load in promille (0-1000). This value is updated at approximately every 250ms.

_btruntime	(*AT % M 030A*)	: UDINT (4 byte unsigned);	Background Runtime Counter
------------	-----------------	----------------------------	----------------------------

This variable contains the background runtime counter. The counter is decremented in 10 ms steps starting with the programmable value. If the value is 0, the CPU triggers a background runtime error (Error 68). The counter is reloaded with the programmable value for every program cycle.

_swbtruntime	(*AT % M 030E*)	: UDINT (4 byte unsigned);	Set point for the Background Runtime Counter
--------------	-----------------	----------------------------	----------------------------------------------

This variable contains the programmable value for the background runtime counter and can be changed during runtime. Set this variable to 0 in order to disable the background runtime counter. The default setting is off (0).

_runstatus	(*AT % M 0EAA*)	: USINT (1 byte unsigned);	Current run status.
------------	-----------------	----------------------------	---------------------

This variable contains the current error value of the CPU in the status display as well as in the status line of LASAL software.

- | | |
|------|----------------------------------|
| 0 | Program was started from the RAM |
| 1 | Program was started from the ROM |
| 2 | Runtime error |
| 3 | Pointer |
| 4 | Checksum error |
| etc. | |

_ClockTicks	(*AT % M 0EAB*)	: UINT (2 byte unsigned);	System-clock in μ s
-------------	-----------------	---------------------------	-------------------------

This variable contains the system clock cycle in μ s.

_runtime	(*AT % M 0EAE*)	: USINT (1 byte unsigned);	Runtime counter.
----------	-----------------	----------------------------	------------------

This variable contains the runtime counter. The counter is decremented in 10 ms steps starting with the programmable value. If the value is 0, the CPU triggers a runtime error (Error 02). The counter is reloaded with the programmable value for every program cycle.

_swruntime	(*AT % M 0EAF*)	: USINT (1 byte unsigned);	Set point for the Runtime-error counter.
------------	-----------------	----------------------------	------------------------------------------

This variable contains the programmable value for the runtime counter. The standard setting is 255.

_error_cnt_dias	(*AT % M 0E94*)	: USINT (1 byte unsigned);	Error counter for the DIAS bus.
-----------------	-----------------	----------------------------	---------------------------------

This variable contains the current counter position of the DIAS bus error counter. Every re-entry attempt to the DIAS bus increases the counter value. The counter is constructed as an 8-bit endless counter.

_FirstScan	(*AT % M 0EB8*)	: USINT (1 byte unsigned);	High on program initialization.
------------	-----------------	----------------------------	---------------------------------

The first scan is a byte address in the data memory and is set at 1 for only the duration of the first program cycle after the operating systems has started the program.

_IOsegment	(*AT % M 0EBA*)	: UDINT (4 bytes unsigned);	IO segment address.
------------	-----------------	-----------------------------	---------------------

This variable contains the segment address of the DIAS periphery. This is required to execute direct address accesses on the DIAS function module.

_DIASconfig	(*AT % M 0EC0*)	: ARRAY [0..63] OF USINT (64 1-byte fields unsigned)	Contains DIAS hardware IDs for each possible DIAS station number.
-------------	-----------------	------------------------------------------------------	-------------------------------------------------------------------

This array contains the system configuration of the DIAS system. Every byte represents a station number (0 bytes...Station 00, 63 bytes...Station 63) in the system. The value contained in a byte identifies the DIAS card at a given station number.

_CDIASconfig	(*AT % M 0f00*)	: ARRAY [0..7] OF USINT (1-byte fields unsigned)	Contains C-DIAS hardware IDs for each possible DIAS station number.
--------------	-----------------	-----------------------------------------------------	---------------------------------------------------------------------

This array contains the system configuration of the C-DIAS system. Every byte represents a station number (0 bytes...Station 00, 7 bytes...Station 7) in the system.

The value contained in a byte identifies the C-DIAS card at a given station number. This is available starting with LasalOS Version 01.01.056

_CpuDisplay	(*AT % D 0001*)	: USINT (1 byte unsigned)	Display on x 386 PLCs.
-------------	-----------------	---------------------------	------------------------

The Display variable represents the D 0001 address.

The content of this address is shown as decimal in the CPU display. However, only the values 00-99 can be displayed. The display remains dark for values over 99.

OPS	(*AT % D 00003*)	: OPSys (pointer on struct OPSys)	Real time task data.
-----	------------------	-----------------------------------	----------------------

2.1.1 Content of the OPS Structure

```

tAbsolute: UDINT (4-byte unsigned) Absolute time from start in ms.
SysState: CONFSTATES      LASAL runtime status.

ConfigTime: UDINT;        Configuration time.

OverRun: UINT;            Counts overrun conditions.

RtInterv_mSec: UDINT;    Always set at 1.
RtInterv_uSec: UDINT;    Always set at 1000.

udDescCRC: UDINT CRC of the active project, calculated from all descriptor lists.

uiLoaderVersion : STRUCT Loader version entry.
usLoRev: USINT;
usHiRev: USINT;
        END_STRUCT;
END_STRUCT;

```

_Isl_pOS	(*AT % M 0EA2*)	: ^LSL_OSDATA (pointer on OS interface structure)	_Isl_pOS
----------	-----------------	---------------------------------------------------	----------

This is the LASAL API, via which the RTOS library also accesses the LASAL OS functions. See RTOS-lib documentation for more information.

_UserHeapStartAddr	(*AT % M 0020*)	: UDINT (4 bytes unsigned)	start address of UserHeapMemory
_UserHeapTotalSize	(*AT % M 0024*)	: UDINT (4 bytes unsigned)	total size of UserHeapMemory
_heapAllocCnt	(*AT % M 0312*)	: UDINT (4 bytes unsigned)	number of UserHeap allocations
_UserHeapUsedMem	(*AT % M 0028 *)	: UDINT (4 bytes unsigned)	amount of used memory from UserHeap

This variable contains the amount of used memory and the internal memory management information. For this reason if 100 bytes of user heap are allocated, for example, then over 100 bytes of memory are used.

_UserHeapFreeMem	(*AT % M 002C*)	: UDINT (4 bytes unsigned)	amount of free memory from UserHeap
------------------	-----------------	----------------------------	-------------------------------------

This variable contains the sum of all free memory blocks. If e.g. 100 bytes of memory are free, the allocatable memory size is less than this value.

_MaxDataMem	(*AT % M 003C*)	: UDINT (4 bytes unsigned)	maximum amount of the projects data memory
-------------	-----------------	----------------------------	--------------------------------------------

This variable contains the maximum amount of used memory for system variables and global variables.

_MaxCodeMem	(*AT % M 0040 *)	: UDINT (4 bytes unsigned)	maximum amount of the projects code memory
-------------	------------------	----------------------------	--------------------------------------------

This variable contains the maximum amount of the code memory available for the application.

_WhoAml	(*AT % M 02E8*)	: UDINT (4 bytes unsigned)	Information about CPU type
---------	-----------------	----------------------------	----------------------------

This variable contains the CPU type in terms of an integer value. The definition and assignment of the CPU type can be found in the `lsl_st_kernel.h` file.

_S_RAM_Hptr	(*AT % M 0300 *)	: ^MRAM_DESCR (pointer on MRAM_DESCR structure)	pointer to S_RAM_Header
-------------	------------------	-------------------------------------------------	-------------------------

This structure contains such information about the SRAM as the data start address, data length and the amount of used data.

_RTrtlntVal	(*AT % M 0306 *)	: UDINT (4 bytes unsigned)	variable not used
-------------	------------------	----------------------------	-------------------

_OnlineMap	(*AT % M 0E96 *)	: ^_OnlineMap (pointer on _OnlineMap structure)	counter of online users
------------	------------------	----------------------------------------------------	-------------------------

2.1.2 Content of the _OnlineMap Structure

bySerial	USINT	Serial online counter
byTCP	USINT	TCP online counter
byCAN	USINT	CAN online counter
byReserve	USINT	not used

2.2 API MultiTask

2.2.1 General Instructions

2.2.1.1 Multitasking

What is Multitasking?

A thread, also called a task, is a sequential path for processing instructions. The discrete statements of a thread are sequentially processed according to the syntactic structure of the C, C++ or Pascal. The term multitasking means that multiple sequential tasks are processed in parallel. In single-processor systems however, it is not possible to run multiple tasks at the same time. So task changes have to be executed. This is the job of a multitasking system such as the LASAL OS. In many cases, tasks cannot be fully processed independently from one another. Despite this, they are expected to work together. For example, a specific task can be required to continue running after another task has completed a specified operation. In this case the according tasks have to be synchronized. E. g.: If parallelism of tasks is limited, synchronization can be implemented with inter-task communication. For a clear understanding of parallel programming, it is essential that one is familiar with the different requirements of multitasking systems. Both of the following sections describe the most important changes between timesharing and real-time systems.

Timesharing

Timesharing uses multitasking to enable simultaneous sharing a high-performance computer by multiple users (or batch jobs). Normally, independent timesharing systems process tasks. Inter-task communication is therefore only available in a simple

configuration. One of the most popular timesharing systems is Unix, which was developed when the performance of available computers was lower than the requirements of the jobs. As a result, the multitasking system limited the computing power as fair as possible among competing tasks. What should be noted is that since allocation required significant resources, processing was normally degraded through multitasking in such systems. Under Unix for example, it is possible for two computer-connected programs to each run one minute long when it is the only active program. If the run parallel, the would need 2.5 minutes. Parallel processing can therefore be more fair, since user 1 and 2 must both wait equally long until the tasks are complete. The form of processing is normally less efficient.

Real-Time Systems

Real-time systems meet completely different requirements. A real-time system can never be overloaded. As soon as there are no more resources available, real-time operation is no longer supported. For example, the following situation could arise with real-time requirements: A counter generates data every second, which the computer must then collect, process and store. If processing a data set requires more than one second in the target computer, the system is overloaded and real-time operation can no longer be supported. Real-time systems are not concerned with "fairness". Tasks can have priorities, which must be strictly followed. A task with a higher priority can take the CPU time of another task with a lower priority at any time; without having to treat the other task "fairly". Since overloading is not allowed, low-priority tasks will possibly receive less CPU time. An additional property of real-time systems is that they must react to events within a predefined (or detect when it is not timespan needed), which is normally short. External events are, when possible, processed using interrupts. Interrupt handlers therefore have the strictest real-time requirements. For this reason, real-time system must have a low interrupt latency time (this is the time between the interrupt signal and running the first interrupt handler). For the function of the tasks, it is essential to differentiate between cooperative and preemptive scheduling. In preemptive scheduling, the reaction time of the task to interrupts is the same as the interrupt latency and switching time of the task. In cooperative scheduling, the maximum timespan between two calls of the kernel is added.

Cooperative and Preemptive Multitasking

Preemptive multitasking means that the task change can be triggered by the interrupt handler. With cooperative (non-preemptive) multitasking, the task change is only made when a task calls the kernel. This means that it behaves "cooperatively" and gives the kernel a chance to change tasks. Example: A receive-interrupt handler for a serial port writes data to a mailbox. When a task is waiting for the mailbox, it is immediately activated by the scheduler with preemptive scheduling. With cooperative scheduling, the task is now set to the "Ready" status. A task change is not performed immediately. After the interrupt

handler is complete, the interrupted task continues. This type of "pending" task change is later run via the kernel as soon as it is called by the active task. LASAL OS supports cooperative, as well as preemptive schedule. The multitasking function is set to cooperative planning by default.

Real-Time

The term "real time" is often used, but seldom defined. One possible definition is:

Real-Time software can synchronously operate with events outside of the processor. The maximum reaction time for external events is predictable. This definition however, says nothing about multitasking. Multitasking is not necessarily required to develop real-time software. However, multitasking can simplify the development of real-time software. Multitasking can achieve excellent reaction times, also when other jobs must be run parallel to real-time processing. The requirements for real-time processing are a sufficiently short interrupt latency, a constant task-change time that should be as short as possible and (in many cases) preemptive scheduling. For this reason, complex operations with non-deterministic runtime requirements such as loading a process from the hard drive, are not supported by a real-time system while performing a task change.

LASAL OS Scheduler

The scheduler decides, which task is executed. In the regard, only task statuses and priorities are considered. Contrary to the many other systems, LASAL OS is not primarily timer-interrupt based. Instead, LASAL OS is an event-driven system. An event is inter-task communication, which can be initiated through a task or interrupt handler and can lead to status changes in the tasks involved. The LASAL OS timer interrupt handler is only one interrupt handler among many. Its purpose is to change tasks to the "Ready" status at a specified time. Several multitasking programs can operate completely without the timer interrupt.

The scheduler operates by the following rules:

1. The task with the highest priority is executed before all others.
2. If the scheduler has the choice between multiple Ready tasks with the same priority, the task that has been waiting the longest is activated.
3. If multiple several tasks are waiting for an event, they are activated according to their priority when the respective event is triggered.
4. With the exception of the time-slice task switches, task changes are only performed when rule 1 would be violated. The number of task changes is therewith minimized.

Normally, the above rules are followed without exception. Whenever a task status changes, the scheduler checks whether a task change should be made according to the

schedule rules. Only with cooperative scheduling, can rule 1 be violated between an interrupt and the next call to the kernel.

Changing Tasks

There are three important types of task changes in LASAL OS: Blocking, activation and time slice. A task always continues at the point where it was stopped before a task change. Under the following conditions, three types of task changes are triggered.

Blocking task change	A blocking task is triggered when a task blocks itself and cannot be run further. This can for example, occur through an attempt to call data from an empty mailbox or if a task releases CPU time for a certain time via calling the TASKDELAY functions. In this case, LASAL OS will run the task with the highest priority that is in Ready status (scheduling rules). If none of the tasks are ready, the idle task is activated.
Activating task changes	Task changes are triggered when a task with a higher priority than the current task is in Ready status. A task change is for example, triggered when a higher priority task is waiting for data from a mailbox and another has stored data at that location. The waiting task cannot continue with calling the data and is immediately activated.
Time slice task change	The LASAL OS timer interrupt handler performs a time slice task change when the above conditions are met. Cooperative time slice task changes can be triggered by calling the TASKDELAY (0) function directly.

In addition to the types of task changes described above, a distinction is made between preemptive and cooperative task changes. Preemptive task changes are triggered through an interrupt handler, while cooperative task changes are initiated by tasks. Blocking tasks changes are always cooperative, since they cannot be initiated by interrupt handlers. LASAL OS sees a preemptive, blocking task change as an error. The activation of tasks using this mechanism can occur in both forms. In the first case, the task change is cooperative. In the last, it is preemptive.

LASAL OS can be configured for cooperative or preemptive scheduling. If preemptive scheduling is deactivated, preemptive task changes from the kernel are delayed until the active task calls a scheduler function (a LASAL OS API function). Potential preemptive task changes are therefore converted into cooperative task changes when the application does not have preemptive scheduling activated.

Time slice task changes are made when the following conditions are met:

1. Time slicing must be enabled (this setting is disabled by default)
2. At least one task with the same priority as the active task must be in Ready status.
3. The last task change must have occurred at a time point specified by the time slice. The last task change was triggered another event as time slicing.

LASAL OS sets the time slice value. For time slicing, preferences are not required. Time slicing plays only a small role and for this reason, cannot violate the scheduling rules described above.

LASAL OS Tasks / Threads

A task or thread is a C or structured text function, which has its own stack. Task priority stages (for the application) range from 1 to 4. A high-priority task means a high urgency for its processing. Priorities only make sense in relation to one another. For example, the function The tasks are referenced with task handles. Task handles are similar to file handles. When a task is created, LASAL OS provides a unique handle that can later be used as a reference (e.g. send data). With the exception of static variables, all local task variables are stored in the stack. The same applies for the local variables for all functions, which are called by the task. Multiple tasks can therefore be started with the same task (initiating) function. Each task is allocated its own stack, as well as its own local variables. A single function can be called via different tasks, by which the same code is used. Since each stack has its own stack, no reentry problems occur as long as the function accesses its own parameters and local (not-static) variables. The visibility guidelines for C / Structured text All tasks can access global data. When the application is initialized, three threads are created: The RTWORK and CYCLE and BACKGROUND task. There, the LASAL tasks are processed.

A task always has one of the following status:

Current	The active task is in Current status as it is being run. In LASAL OS, there can only be one task at a time in this status.
Ready	All tasks that are ready to start, are set to Ready status. Normally, all Ready tasks have either the same or a lower priority as the active task.
Suspended	Suspended tasks cannot be run, because they were stopped by LASAL OS with the SUSPEND function. In order to run suspended tasks, the LASAL OS Resume function must be called.
Blocked	Tasks that are blocked cannot be run, because they are waiting for an event (e.g. a semaphore signal or an incoming message in the mailbox). These tasks can only be reset by another task or an interrupt handler.
Delaying	These tasks have blocked themselves for a specific timespan. The LASAL OS timer interrupt handler automatically resets them after the delay time has elapsed.
Timed	Timed tasks wait for an event or a timeout. Such tasks are reset to Ready when the event occurs or a timeout is triggered.

LASAL OS contains all inactive tasks in several queues. For example, there is a task queue for all Ready tasks and another queue contains all tasks, which wait for a specified time. Queues are also formed with semaphores or mailboxes when tasks there block.

LASAL Tasks

LASAL tasks are methods of a LASAL object (RtWork, CyWork or Background), which are cyclically called by the operating system. If the application is initialized, the loader writes the LASAL task in the task list. Task lists are for real-time, cyclic and background tasks. Each task list entry contains a time interval (period), which can be configured in LASAL classes (or objects).

The application tasks are processed in an infinite loop in order to check every entry (object), as to whether the time interval has elapsed since the last call. If the specified time has elapsed, the method is called and the time point of the last call is updated.

After the task list is complete (application ended / in Reset / in Error), the application tasks are allowed to continue for s specified time in order to process the task list. The real-time task is run for a time period of 1 ms in IPCs and 2 ms for 386 CPUs. With current platforms, this value is configured (Tick: 125µs – 2ms). The time period for the cyclic task is a minimum of 1 ms, but can be higher when the tick is higher. The background task is normally 2 ms.

Inter-task Communication

The term inter-task communication encompasses all components, which are used for information exchange between tasks. LASAL OS provides three different techniques: semaphores, mailboxes and message passing

Practically all multitasking systems provide semaphore s, which are used for signal exchange to activate or block tasks. A semaphore is a variable in which a signal value can be stored or read. Task changes can occur when a semaphore with a value of 0 is accessed. There are five different semaphore types in the LASAL operating system: counter, binary, event, resources and mutex.

Mailboxes are an expansion of the semaphore concept. Instead of signal values, data can be stored or read from a mailbox. A task change occurs when an accessed mailbox is empty or completely full. The number of data sets in a mailbox can be configured. Mailboxes are especially useful for buffering data between tasks or interrupt handlers and tasks.

Message passing is used of data exchange directly between two tasks; no data or signals are buffered. Since the tasks must synchronize, it is the closest coupling between tasks.

Reentrance

The term reentrance refers to problems that can occur when several tasks run the same code or access global data at the same time. Reentrant code means that a task can

process the code before another task has finished running it. In a multitasking system, it is important to ensure that as much code as possible is reentrant, so that it can be used by multiple tasks.

The following example demonstrates the problems that can occur when global data are used. It is assumed that two tasks are used for counting and the program contains a global "Counter" variable of type int, which is initialized with a value of 0. Both tasks run will run the execute instructions.

```
Counter = Counter + 1;
```

The counter always counts when an event occurs.

The compiler could translate this command as follows.

```
MOV EAX, Counter;      line 1  
ADD EAX, 1;           line 2  
MOV Counter, EAX;    line 3
```

For simplification, it is assumed that both tasks have the same priority, preemptive scheduling activated and the time slicing is enabled.

The following could occur: Task 1 detects an event and begins processing the machine language instructions above. After line 1 was run (register EAX contains 0), a time slice task change is triggered and task 2 is activated. Task 2 also detects an event and increments the counter variable by 1 without being interrupted. Later, task 1 (in line 2) increments EAX again from 0 to 1, and stored this value in the variable Counter.

Even when Counter was increased twice, it still only has a value of 1.

This example shows that two or more tasks can never access global data at the same time, when at least one of tasks can change the data. In reality, it is not the code that is reentrant, but the data manipulated by the code. Even when in both tasks, the statement "Counter = Counter + 1;" is separate, the problem is not solved. For this reason, global data should never be shared. The same is true for local variables declared as static. If sharing global data cannot be avoided, it should be protected by the use of semaphore s. Unfortunately, there are some parts of the code, which the programmer cannot control. For example, run-time system libraries.

2.2.1.2 Time

For real-time systems, the time is of great importance. LASAL OS has an interrupt-controlled clock, which can be set as well as read. In addition, a task can block itself for a specified time to free CPU time for other tasks. The time is expressed in timer ticks. In LASAL CLASS, the time is expressed in milliseconds. LASAL OS receives it timer interrupts from the clock device driver. This is only used to periodically generate interrupts. LASAL OS does not know how much time (e.g. in seconds) elapses between two sequential timer ticks.

2.2.1.3 Message Passing

In addition to the mailboxes, LASAL OS provides a further mechanism for inter-task communication with message passing. With message passing, data objects such as semaphore s or mailboxes for data buffering are no longer required. Data are copied directly between the tasks. Message passing is comparable to a mailbox with a size of 0.

The basic difference is that the send task communicates with the receive task directly (addressed via handle). A receive task cannot however, distinguish between the tasks from which data is received. With mailboxes, every task can use any mailbox. However, a send task cannot determine who receives the data and the receive task does not know who sent the data.

Another difference is, that there is no data buffering. Before the data can be copied, the send and the receive tasks must be ready for the transfer. When two tasks want to exchange data via message passing, the first task is blocked as soon as it has reached the send or receive status, until the second task has reached its corresponding receive or send status. The data transfer then is finished and the tasks can be continued.

Mailboxes do not provide such a narrow connection. For example, a task can continue running immediately after an operation; even if there is no task that can receive data. Message passing can only be used for synchronization. In this case, the receive task defines a data length of 0 and the pointer to the data can have a value of NULL, because LASAL OS does not perform a transfer data.

Due to the coupling between tasks and the missing data buffer, this mechanism is normally not used with interrupt handlers. However, it is possible.

With mailboxes, ensure the mailbox used must be sent as an input parameter. With message passing, define the receiver (task handle) for the send task. Since the receive task does not know how sent the data, it does not define the source.

2.2.1.4 Semaphore Handling

Semaphores are popular tools for synchronizing flag. A semaphore can be seen as an event counter, which can never receive a negative value and can be used by any task with the functions described in this section. A program can use any number of semaphores.

The SIGNAL function stores an event in a semaphore, WAIT then calls an event from the semaphore when it is available. When no event is available, it waits until an event occurs. There are five types of semaphores in LASAL OS: Counter, binary, event, resources and mutex semaphores. The semaphore type is defined when creating the semaphore.

Counter semaphores can store up to $2^{32}-1$ events and are useful for synchronization and correspond to the Dijkstra or Ben-Ari definitions. Binary semaphores cannot be used as

counters The SIGNAL function always sets a binary semaphore to 1, although it already has received a value of 1. The WAIT function sets the value to 0.

Event semaphores are similar to binary semaphores with the exception, that the WAIT function does not decrement the counter value. A single call to the SIGNAL function can set any number of tasks waiting for the event semaphore to Ready. To reset an event semaphore to 0, the RESETEVENT or PULSE function must be called. These functions are only available for event semaphores. The SIGNAL function sets the event semaphore to the value 1; the WAIT function has no effect on the value of the event semaphore.

Resource semaphores are for resource management. A resource semaphore ensures that the priority of a task with a resource semaphore is at least the highest of all other tasks, which require the respective resource. This technique is called priority inheritance. A task, which requests a resource via the Wait function, gives its priority to the occupying resource task. This ensures that a task with a higher priority is never unnecessarily blocked by a lower priority task.

Priority inheritance can be seen as pseudo priorities for resources. For example: A task would like to occupy a resource semaphore, but is blocked because the resource is already set. The task gives its priority to the resource semaphore, which then gives it to the occupying task. A priority can only be inherited when the priority of the resource-occupying task is lower than the task that is waiting. The priority of a task can only be raised; it can never be lower than the base priority. Resource semaphores also allow save termination and suspending of tasks. A task can only be immediately suspended or ended when it does not occupy a resource semaphore. Otherwise, it continues until it has released all resources and is later suspended or ended. This tool serves to avoid dead locks. For resource semaphores, two limitations apply: A resource must always be released via the SIGNAL function by the task that acquired it through the WAIT function.

A task can occupy any number of resources simultaneously. The resources must however, be released in exactly the reverse order in which they were requested.



```
OS_MT_Wait(R1);  
OS_MT_Wait(R2);  
...  
OS_MT_Signal(R2);  
OS_MT_Signal(R1);
```

If the sequence of the OS_MT_Signal calls is switched, an error is triggered in the LasalOS.

Due to the limitations, resource semaphores are not suited for the use of interrupt handlers.

Mutex semaphores are a variation of resource semaphores. The only difference is that a task can request a resource that it already possesses.



```
OS_MT_Wait(R);
OS_MT_Wait(R);
...
OS_MT_Signal(R);
OS_MT_Signal(R);
```

If R is a resource semaphore, the second WAIT function triggers an error since a tasks cannot acquire a resource it already possesses. For a mutex semaphore however, this construct is legal and the resource is not released until the number of SIGNAL function calls is the same as those of the WAIT function. The disadvantage of the Mutex semaphore as compared to resource semaphores is that unequal WAIT/SIGNAL pairs are not detected. If a call to the SIGNAL function fails is missing, the resource is not released and no error is triggered, even if the WAIT/SIGNAL function continues to call the same mutex.

2.2.1.5 Mailbox Handling

The previous section describes semaphores, which can be used to synchronize tasks and provide a mechanism for orderly inter-task communication with global data. Task communication via global data is difficult to follow and error-prone however, as tasks that require semaphore operations can "forget". In addition, no protocol to monitor data exchange is kept. Mailboxes that are data buffers, which can save a fixed number of messages, are used to close the

In LASAL OS, messages can have any size and the mailbox size can be configured accordingly. Tasks can save messages in a mailbox. If the mailbox is full, the task is blocked until space is available again. Tasks can also call messages from mailboxes. In this case, if the mailbox is empty, the task is also blocked. The same mailbox can be used to store and retrieve messages from different tasks. Messages can be added to a mailbox using [PUT\(\)](#), [PUTCOND\(\)](#) and [PUTTIMED\(\)](#) functions

Reports can also be inserted at the beginning of a message thread with the [PUTFRONT\(\)](#), [PUTFRONTCOND\(\)](#) and [PUTFRONTTIMED\(\)](#) functions. If messages are stored in or called from a mailbox via the [PUT\(\)/GET\(\)](#) function pair, the mailbox operates like a FIFO buffer (first in, first out). This means that messages are called from a mailbox in the same order in which they were stored. If [PUTFRONT\(\)/GET\(\)](#) function pairs are called however, the mailbox operates like a LIFO buffer (last in, first out). For a high degree of flexibility, the [PUT\(\)](#) and [PUTFRONT\(\)](#) functions can be freely combined (also for the same mailbox).

2.2.2 Multitask Class: Interface Functions

2.2.2.1 Functions

Extended

GETLASTERROR	GETTIME		
------------------------------	-------------------------	--	--

Task Functions

CREATETHREAD	CURRENTTASKHANDLE	DELAYUNTIL	GETMINSTACK
GETTASKPRIORITY	GETTASKSTACK	GETTASKSTATE	RECEIVE
RECEIVECOND	RECEIVETIMED	RESUME	SEND
SENDCOND	SENDTIMED	SETPRIORITY	SUSPEND
TASKDELAY	TERMINATETASK		

Semaphores

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 Extended Functions

2.2.2.2.1 GETLASTERROR

Returns the last error code.

```
FUNCTION __cdecl virtual global GETLASTERROR
VAR_OUTPUT
    ret0      : DINT;
END_VAR;
```

Transfer parameters	Type	Description
none		
Return parameters	Type	Description
ret0	DINT	Last error code

The OS interface functions either return an error code (DINT value) or the value NIL. GETLASTERROR() returns the last error code of a function.



The following codes are implemented:

MERROR_NONE	No error
MERROR_NOMEM	Not enough memory to execute this function, e.g. internal allocation failed
MERROR_NOFCT	CreateTask needs a valid task function
MERROR_FCTNOTINMEM	CreateTask needs a task function placed in the LASAL code memory
MERROR_WRONGPRIOR	CreateTask only supports priorities from 1 to 14
MERROR_STACK	CreateTask supports stack sizes up to 0x4000
MERROR_NAME	CreateTask, CreateMailbox or CreateSemaphore need a valid name
MERROR_NAMEUSED	CreateTask, CreateMailbox or CreateSemaphore name already used
MERROR_HANDLE	Handle OS list => invalid
MERROR_NOTALLOWED	MT function not allowed in this TaskContext
MERROR_MESSAGEGESIZE	Mailbox needs a valid message size

MTEROR_DATA	Pointer to the data is not valid
-------------	----------------------------------

2.2.2.2.2 GETTIME

This function can be used to read the LASAL OS kernel clock.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETTIME
VAR_OUTPUT
    ret0      : DINT;
END_VAR;
```

Transfer parameters	Type	Description
none		
Return parameters	Type	Description
ret0	DINT	Current time of the LASAL OS internal kernel clock in milliseconds

GETTIME() returns the number of timer pulses since the start of the LASAL OS (if no overflow occurred).



2.2.2.3 Task Functions

2.2.2.3.1 CREATETHREAD

With this function each task in a program can create other tasks/threads.

```
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;
```

Transfer parameters	Type	Description
taskfunction0	pVoid	A LASAL function with __cdecl or conventional LASAL CLASS call convention and one single pVoid parameter (VAR_INPUT); this

		contains the code to execute of the new task (entry function)																		
priority0	UDINT	<p>The base priority of the new task as an integer between MT_MIN_PRIORITY (1) and MT_MAX_PRIORITY (14)</p> <p>The table below shows an overview of the standard tasks of LASAL OS</p> <table border="1"> <thead> <tr> <th>Thread (Task)</th> <th>Task</th> <th>Priority</th> </tr> </thead> <tbody> <tr> <td>Real-Time</td> <td>Processing of the RtWork LASAL tasks</td> <td>16</td> </tr> <tr> <td>OS Kernel task</td> <td>Runtime monitoring, executing operating system functions</td> <td>15</td> </tr> <tr> <td>Cyclic</td> <td>Processing of the CyWork LASAL tasks</td> <td>14</td> </tr> <tr> <td>Communication tasks</td> <td>Online communication</td> <td>14</td> </tr> <tr> <td>Background</td> <td>Processing Background LASAL tasks The priority of the background task depends on the SET VISU LOW HIGH setting (LOW: 10, HIGH: 14)</td> <td>10/1 4</td> </tr> </tbody> </table>	Thread (Task)	Task	Priority	Real-Time	Processing of the RtWork LASAL tasks	16	OS Kernel task	Runtime monitoring, executing operating system functions	15	Cyclic	Processing of the CyWork LASAL tasks	14	Communication tasks	Online communication	14	Background	Processing Background LASAL tasks The priority of the background task depends on the SET VISU LOW HIGH setting (LOW: 10, HIGH: 14)	10/1 4
Thread (Task)	Task	Priority																		
Real-Time	Processing of the RtWork LASAL tasks	16																		
OS Kernel task	Runtime monitoring, executing operating system functions	15																		
Cyclic	Processing of the CyWork LASAL tasks	14																		
Communication tasks	Online communication	14																		
Background	Processing Background LASAL tasks The priority of the background task depends on the SET VISU LOW HIGH setting (LOW: 10, HIGH: 14)	10/1 4																		
stackSize0	UDINT	Size of the stack allocated for the new task (in bytes); the lower limit is 512 bytes, the upper limit 16 kB (16384 Bytes / 0x4000)																		
flags0	UDINT	Can be used, to select options for the new task; description see below																		
parameter0	pVoid	Parameter transferred to the entry function of the task or the this pointer (see semaphore s0)																		
name0	^char	Pointer to the name of the task; LasalOS inserts USR_ at the beginning of the name.																		
Return parameters	Type	Description																		
ret0	MT_TASKHANDLE	LASAL MT Task handle (MT_TASKHANDLE); this is a reference to the newly created task																		

In taskfunction0 (entry function), the same function can be used multiple times to create tasks/threads. These tasks will then execute the same code, but each will have its own local data. Tasks/threads created with [CREATETHREAD\(\)](#) are always called only once. When the execution reaches the end of the function, the thread is deleted (runs out). To remain in the thread, a loop (versatile loop) must be implemented that is continuously run (attention: formulate an exit condition – see [TERMINATETASK\(\)](#)).

The task priority (priority0) indicates the urgency. The higher the priority of the task, the sooner it is processes. If the priority of the new task is higher than that of the active task, the new task is activated (when MT_TASK_SUSPENDED is not set as a flag). LasalOS distinguishes between the base priority and the processing priority. For the scheduling (task change) only the execution priority is of interest. The processing priority of a task is generated from the level of its base priority and the priorities of all (resource and mutex) semaphores, which are occupied by the task. The priority of a (resource or mutex) semaphore is generated from the highest priority of the task waiting for the semaphore. This procedure is called priority inheritance.

LASAL OS needs at least 512 bytes stack per thread. If the parameter stackSize0 is smaller than 512, 512 (bytes) is used. It is important to note that stack overflows are among the most frequent and difficult to determine errors in multi-tasking systems. During program development, the [GETTASKSTACK\(\)](#) and [GETMINSTACK\(\)](#) functions should be used to thoroughly analyze the current stack usage of the tasks. With the design phase, the memory space for the stacks should be dimensioned generously (8-16 k).

At the moment the following values are defined for the parameter semaphore s0.

MT_TASK_SUSPENDED	If this semaphore is set, the task is created in suspended mode. It is not started, until RESUME() is called.
MT_TASK_MATH_CONTEXT	This flag instructs LASAL OS to retain a floating decimal context for the new task. This semaphore cannot be combined with TF_NO_MATH_CONTEXT.
MT_TASK_NO_MATH_CONTEXT	This flag instructs LASAL OS not to retain a floating decimal context for the new task. The semaphore cannot be combined with TF_MATH_CONTEXT.
MT_TASK_SAVETHIS	This flag instructs LASAL OS to initialize the ESI register with the value in parameter0 before running the task. Parameter0 must contain the This pointer.

If neither **MT_TASK_MATH_CONTEXT** nor **MT_TASK_NO_MATH_CONTEXT** are specified, LasalOS creates a floating point decimal context by default. The parameter parameter0 is transferred to the function of the task. LasalOS does not interpret this value; it is simply forwarded. It can be used to transfer any information to the new task.

The new task name (name0) is for simple identification and should not be longer than 15 characters (and cannot be longer than 32). LasalOS inserts **USR_** at the beginning of the name string. The name may only be used once.

2.2.2.3.2 CURRENTTASKHANDLE

Returns the handle of the currently executed task.

```
FUNCTION __CDECL VIRTUAL GLOBAL CURRENTTASKHANDLE
VAR_OUTPUT
    ret0      : MT_TASKHANDLE;
END_VAR;
```

Transfer parameters	Type	Description
none		
Return parameters	Type	Description
ret0	MT_TASKHANDLE	Handle of the currently active task or NULL when the function is called in the LASAL TASK layer: RTWork, CyWork, etc. ..

2.2.2.3.3 DELAYUNTIL

Task that should continue at a certain time, can use DELAYUNTIL().

```
FUNCTION __CDECL VIRTUAL GLOBAL DELAYUNTIL
VAR_INPUT
    timeout0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
timeout0	UDINT	Indicates the time in milliseconds until the task continues
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code

DELAYUNTIL() can be used, to set cyclic tasks that run in a fixed time frame.

2.2.2.3.4 GETMINSTACK

Provides the number of bytes least available in the stack of a task since its creation.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETMINSTACK
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_TASKHANDLE	Handle of the task from which to request the stack
Return parameters	Type	Description

ret0	DINT	Minimum available space of the stack or -1, if an error occurred
------	------	------------------------------------------------------------------

2.2.2.3.5 GETTASKPRIORITY

Returns the current execution priority of the task.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETTASKPRIORITY
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters		Type	Description
handle0		MT_TASKHANDLE	Handle of the task from which to request the priority
Return parameters		Type	Description
ret0		DINT	Between MIN_PRIORITY and MAX_PRIORITY, if no error occurred or -1 for an error

2.2.2.3.6 GETTASKSTACK

Returns the available stack of a task.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETTASKSTACK
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters		Type	Description
handle0		MT_TASKHANDLE	Handle of the task from which to request the stack
Return parameters		Type	Description
ret0		DINT	Free stack or -1, if an error occurred

Can be used to query the stacks of the current task GETTASKSTACK([CURRENTTASKHANDLE\(\)](#)). It can happen, that LasalOS is not able to determine the stack limits. In this case, the value FFFFFFFFh is returned. The actual available stack never can have this value.

2.2.2.3.7 GETTASKSTATE

Returns the actual status of a task. For the current task, the task handle can be retrieved with [CURRENTTASKHANDLE\(\)](#).

```
FUNCTION __CDECL VIRTUAL GLOBAL GETTASKSTATE
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : LSL_MT_TASKSTATE;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_TASKHANDLE	Handle of the task from which to request the state
Return parameters	Type	Description
ret0	LSL_MT_TASKSTATE	Status of the type LSL_MT_TASKSTATE

One of the following values can be returned:

MTASKSTATE_READY	The task is ready for execution.
MTASKSTATE_CURRENT	The task is being executed.
MTASKSTATE_SUSPENDED	The task has been suspended by calling SUSPEND() .
MTASKSTATE_DELAYING	The task is paused by a call to TASKDELAY() or DELAYUNTIL() .
MTASKSTATE_BLOCKED_WAIT	The task is paused to wait for a semaphore by a call to WAIT() .
MTASKSTATE_TIMED_WAIT	The task is paused to wait for a semaphore by a call to WAITTIMED() .
MTASKSTATE_BLOCKED_PUT	The task is paused by a call to PUT() , due to a full mailbox.
MTASKSTATE_BLOCKED_GET	The task is paused by a call to GET() , due to an empty mailbox.
MTASKSTATE_TIMED_PUT	The task is paused by a call to PUTTIMED() , due to a full mailbox.
MTASKSTATE_TIMED_GET	The task is paused by a call to GETTIMED() , due to an empty mailbox.
MTASKSTATE_BLOCKED_SEND	The task is paused to wait for the receiver task by a call to SEND() , in order to indicate that it is in receive status.
MTASKSTATE_BLOCKED_RECEIVE	The task is paused by a call to RECEIVE() , in order to receive data from a send task (see SEND()).
MTASKSTATE_TIMED_SEND	The task is paused to wait for the receiver task by a call to SENDTIMED() , in order to indicate that it is in receive status.

MTASKSTATE_TIMED_RECEIVE	The task is paused by a call to RECEIVETIMED() , in order to receive data from a send task (see SEND()).
MTASKSTATE_DEADLOCKED	The task is blocked by a send operation (message passing), whereby the receiver task was ended.
MTASKSTATE_ILLEGAL	The transferred handle is not based on an existing task.
MTASKSTATE_TERMINATED	The task has self-terminated by a call to TERMINATETASK() with its own handle or its task function was completed. The task can no longer run, but still exists because its memory has not yet been freed.

2.2.2.3.8 RECEIVE

Receives data from another task.

```
FUNCTION __CDECL VIRTUAL GLOBAL RECEIVE
VAR_INPUT
    data0          : pVoid;
    dataLength0    : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
data0	pVoid	Pointer to the variable, where the received data are saved
dataLength0	UDINT	Length of the expected data
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code

If a task sends data via [SEND\(\)](#) or [SENDTIMED\(\)](#) to a receive task blocked in `OS_MT_Receive()`, the data are immediately transferred and the task with the highest priority continues running. Otherwise the receive task is blocked, until another task sends data.

2.2.2.3.9 RECEIVECOND

Receives data from a task.

```
FUNCTION __CDECL VIRTUAL GLOBAL RECEIVECOND
VAR_INPUT
    data0          : pVoid;
    dataLength0    : UDINT;
END_VAR
VAR_OUTPUT
```

```
    ret0      : DINT;
END_VAR;
```

Transfer parameters	Type	Description
data0	pVoid	Pointer to the variable, where the received data are saved
dataLength0	UDINT	Length of the expected data
Return parameters	Type	Description
ret0		TRUE, if the data transfer finished successfully, otherwise an error code

Receives data from each task, if the task is immediately ready to send. RECEIVECOND() never blocks the task.

2.2.2.3.10 RECEIVETIMED

Receives data from a task.

```
FUNCTION __CDECL VIRTUAL GLOBAL RECEIVETIMED
VAR_INPUT
    data0      : pVoid;
    dataLength0 : UDINT;
    timeout0   : UDINT;
END_VAR
VAR_OUTPUT
    ret0      : DINT;
END_VAR;
```

Transfer parameters	Type	Description
data0	pVoid	Pointer to the variable, where the received data are saved
dataLength0	UDINT	Length of the expected data
timeout0	UDINT	Waiting time until the receive task is ready to accept data (in milliseconds)
Return parameters	Type	Description
ret0	DINT	TRUE, if the data transfer finished successfully, otherwise an error code

Receives data from each other task, if the task is ready to send data within the waiting time.

2.2.2.3.11 RESUME

Reactivates a suspended task.

```
FUNCTION __CDECL VIRTUAL GLOBAL RESUME
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters		Type	Description
handle0		MT_TASKHANDLE	Handle of the task to be reactivated
Return parameters		Type	Description
ret0		DINT	MTERROR_NONE, if no error occurred, or an error code

If the task is not suspended, RESUME() has no effect.

2.2.2.3.12 SEND

Sends data to another task.

```
FUNCTION __CDECL VIRTUAL GLOBAL SEND
VAR_INPUT
    handle0      : MT_TASKHANDLE;
    data0        : pVoid;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters		Type	Description
handle0		MT_TASKHANDLE	Handle of the receiver task
data0		pVoid	Points to the data to send
Return parameters		Type	Description
ret0		DINT	MTERROR_NONE, if no error occurred, or an error code

If the receive task is waiting in a [RECEIVE\(\)](#) or [RECEIVETIMED\(\)](#), the data are transferred immediately and the task with the highest priority continues running. Otherwise the send task is blocked until the receive task is ready to accept the data.

2.2.2.3.13 SENDCOND

Sends data to another task.

```
FUNCTION __CDECL VIRTUAL GLOBAL SENDCOND
VAR_INPUT
    handle0      : MT_TASKHANDLE;
    data0       : pVoid;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_TASKHANDLE	Handle of the receiver task
data0	pVoid	Points to the data to send
Return parameters	Type	Description
ret0	DINT	TRUE, if the data transfer finished successfully, otherwise an error code

Data are sent to another task, if the receive task is ready to immediately accept the data. Otherwise the return value shows an error and no data are transferred. SENDCOND() never blocks the task.

2.2.2.3.14 SENDTIMED

Sends data to another task.

```
FUNCTION __CDECL VIRTUAL GLOBAL SENDTIMED
VAR_INPUT
    handle0      : MT_TASKHANDLE;
    data0       : pVoid;
    timeout0    : UDINT;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_TASKHANDLE	Handle of the receiver task
data0	pVoid	Points to the data to send
timeout0	UDINT	Waiting time until the receive task is ready to accept data (in milliseconds)
Return parameters	Type	Description

ret0	DINT	TRUE, if the data transfer finished successfully, otherwise an error code
------	------	---------------------------------------------------------------------------

Data are sent to another task, if the receive task is ready to accept the data within the waiting time.

2.2.2.3.15 SETPRIORITY

Changing the priority of a task. The new priority must be between 1 and 14.

```
FUNCTION __CDECL VIRTUAL GLOBAL SETPRIORITY
VAR_INPUT
    handle0      : MT_TASKHANDLE;
    priority0    : UDINT;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_TASKHANDLE	Handle of the task of which to change the priority
priority0	UDINT	New base priority of the task
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code

Priority0 must be in the range MIN_PRIO (1) to MAX_PRIO (14). After a call to OS_MT_SetPriority(), the priority for execution of the task is reevaluated and if necessary, the scheduler performs a task change.

2.2.2.3.16 SUSPEND

Can be used to deactivate a task.

```
FUNCTION __CDECL VIRTUAL GLOBAL SUSPEND
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_TASKHANDLE	Handle of the task to be deactivated

Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code

After suspending a task, it can be reactivated by calling the Resume function. If the task is already suspended, the SUSPEND() function has no effect. Before the task is actually suspended, LasalOS ensures that it does not occupy any resource or mutex semaphores. If yes, the task continues, until all resources have been released. The call of SuspendTask does not wait until suspending is finished, but immediately continues.

2.2.2.3.17 TASKDELAY

Blocks the called task for the defined time and allows execution of other tasks.

```
FUNCTION __CDECL VIRTUAL GLOBAL TASKDELAY
VAR_INPUT
    timeout0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
timeout0	UDINT	Specifies the time in milliseconds, over which the task must be blocked
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code

If TASKDELAY() is called with $timeout0 \leq 0$, LasalOS checks whether other tasks with same or a higher priority are ready. If a task with a higher priority is found in the MT_Ready (ready) state, it is then activated. If task with the same priority are ready, the task with the longest waiting time is activated.

TASKDELAY(0) can therewith be used to implement round-robin scheduling (also known as cooperative time slicing). Time slicing may not be activated for this purpose.

2.2.2.3.18 TERMINATETASK

Terminates a task.

```
FUNCTION __CDECL VIRTUAL GLOBAL TERMINATETASK
VAR_INPUT
    handle0      : MT_TASKHANDLE;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
```

END_VAR;

Transfer parameters		Type	Description
handle0		MT_TASKHANDLE	Handle of the task to be terminated
Return parameters		Type	Description
ret0		DINT	MTERROR_NONE, if no error occurred, or an error code

The OS_MT_TERMINATETASK function usually is not necessary. A task is finished, when it reaches the end of its entry function. If the application was ended (Reset, Error ...) all user tasks are also ended (threads, which were generated with [CREATETHREAD\(\)](#)).

If an invalid value has been transferred to the function, the program returns an error. Before the task is ended, LasalOS must ensure that the task does not have any other resource or mutex semaphores. If this is the case, the task continues, until all resources have been released. The TERMINATETASK() function does not wait until the task is closed, but returns directly.

The resources (stack etc.) of the task to end are released, provided that taskhandle0 does not reference the current task. The memory of a self-terminated (not ended with TERMINATETASK(), but timed out) task is released with the next call of [CREATETHREAD\(\)](#). Until then the task still exists in terminated mode, but is no longer executed. Ending a task twice must be avoided (self-terminated and terminated by another task), since when TERMINATETASK() is called the second time, the task no longer exists.

2.2.3 Semaphore Functions

2.2.3.1 CREATESEMAPHORE

Creates and initializes a semaphore.

```
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;
```

Transfer parameters		Type	Description
type0		LSL_MT_SEMATYPE	Desired semaphore type

init0	UDINT	Must be valid for the selected semaphore type
flags0	UDINT	Indicates whether and how the function should search for an existing semaphore
name0	^char	Pointer to the name of the semaphore
Return parameters	Type	Description
ret0	MT_SEMAHANDLE	A valid handle or NIL, if the function is faulty



The values for the parameter type0:

- MTSEMATYPE_COUNTING
- MTSEMATYPE_BINARY
- MTSEMATYPE_EVENT
- MTSEMATYPE_RESOURCE
- MTSEMATYPE_MUTEX

For counting semaphores, the init0 parameter must be 0 to $2^{32}-1$, for binary and event semaphores, 0 or 1 and for resource / mutex semaphores, this parameter must be 1.

The values for the parameter flags0:

The value ,0: A new semaphore is created without a condition.

MTSEMACREATE_SEARCH CREATESEMAPHORE is attempting to find a semaphore of the same type0 and name0, which was also created with MTSEMACREATE_SEARCH. Unnamed semaphores are not checked and the name comparison is case sensitive. If a semaphore was found, a handle is returned to this current semaphore and a new one is not created. In this case parameter init0 is ignored.

MTSEMACREATE_FAIL_NOT_FOUND

This flag can be specified in addition to MTSEMACREATE_SEARCH. If no semaphore can be found, no new one will be created. In this case, ret0 returns the value NIL.

The name of the new semaphore is for simple identification of the flag and should not be longer than 15 characters. LasalOS inserts USR_ at the beginning of the name.

2.2.3.2 DELETESEMAPHORE

An existing semaphore is deleted and invalid.

```
FUNCTION __CDECL VIRTUAL GLOBAL DELETESEMAPHORE
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters		Type	Description
handle0		MT_SEMAHANDLE	Handle of the semaphore
Return parameters		Type	Description
ret0		DINT	MTERROR_NONE, if no error occurred, or an error code



No task may wait for the semaphore. The attempt to access a semaphore after it has been deleted, leads to an error and the behavior is undefined.

2.2.3.3 PULSE

Sets all tasks waiting for an event semaphore to ready (MT_Ready) and immediately resets the semaphore.

```
FUNCTION __CDECL VIRTUAL GLOBAL PULSE
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters		Type	Description
handle0		MT_SEMAHANDLE	Handle of the semaphore
Return parameters		Type	Description
ret0		DINT	MTERROR_NONE, if no error occurred, or an error code



If one or more of these tasks have a higher priority than the calling task, a task change is executed. If handle0 does not refer to an event semaphore, an error is triggered.

2.2.3.4 RESETEVENT

Sets the value of an event semaphore to 0.

```
FUNCTION __cdecl virtual global RESETEVENT
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_SEMAHANDLE	Handle of the semaphore
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code



Further calls of [WAIT](#) or [WAITTIMED](#) are blocking. Use [SIGNAL](#) to set an event semaphore to 1. To free all tasks waiting for the event semaphore, without having to set the semaphore, use [PULSE](#).

2.2.3.5 RESOURCEOWNER

Can be used to determine which task currently occupies a resource or mutex semaphore.

```
FUNCTION __cdecl virtual global RESOURCEOWNER
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : MT_TASKHANDLE;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_SEMAHANDLE	Handle of the resource or mutex semaphore to query

Return parameters	Type	Description
ret0	MT_SEMAHANDLE	If a semaphore was released, the return value is NIL, otherwise, the task handle is returned

2.2.3.6 SEMAVALUE

The number of events saved in a semaphore, can be requested here.

```
FUNCTION __CDECL VIRTUAL GLOBAL SEMAVALUE
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_SEMAHANDLE	Handle of the semaphore
Return parameters	Type	Description
ret0	DINT	Number of events or an error code

The function does not lead to a task change.



2.2.3.7 SIGNAL

Saves an event in a semaphore.

```
FUNCTION __CDECL VIRTUAL GLOBAL SIGNAL
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_SEMAHANDLE	Handle of the semaphore to save the event
Return parameters	Type	Description

ret0	DINT	MTERROR_NONE, if no error occurred, or an error code
------	------	------------------------------------------------------



If several tasks wait for the semaphore, the task with the highest priority is set to ready (MT_Ready). If its priority is higher than that of the calling task, it is immediately activated (MT_Current). If the task is of the type MTSEMATYPE_EVENT, all waiting tasks are set to ready (MT_Ready). If no task waits for the semaphore, the event is saved. A counting semaphore can save up to $2^{32} - 1$ events. The application should care, that this limit is not exceeded. Binary and event semaphores ignore additional events, once they already received one. The attempt to set a resource or mutex semaphore to a value > 1 using the SIGNAL function (multiple signal calls) is an error. In this case, the results are unpredictable. If "Semahandle0" is a resource or mutex semaphore, the (execution) priority of the active task is reevaluated according to the priority assignment rules.

2.2.3.8 WAIT

Calls an event from a semaphore.

```
FUNCTION __CDECL VIRTUAL GLOBAL WAIT
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_SEMAHANDLE	Handle of the semaphore containing the event
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code



If no event is available, the calling task is blocked (MT_BLOCKED_WAIT). It can only be reactivated if another task calls the function SIGNAL for the according semaphore (MT_Ready). Any number of tasks can wait for the events of one semaphore. The tasks are processed in the sequence of their priorities (MT_CURRENT).

If "semahandle0" is an occupied resource or mutex semaphore, the (execution) priority of the tasks that occupy the semaphore is raised to that of the blocking task, if it has a higher priority (priority inheritance).

After calling WAIT, the active task possesses or occupies the resource or mutex semaphore. It cannot be suspended or terminated until it has released all of its resources.

Event semaphores are an exception. After finishing the operation, they decrement the value of the semaphore with WAIT.

2.2.3.9 WAITCOND

Calls an event from a semaphore, if it exists.

```
FUNCTION __CDECL VIRTUAL GLOBAL WAITCOND
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
handle0	MT_SEMAHANDLE	Handle of the semaphore containing the event
Return parameters	Type	Description
ret0	DINT	TRUE, if a signal has been called, otherwise an error code is returned



Calls an event from a semaphore, if it is available immediately. WAITCOND() never leads to a blocking task change.

2.2.3.10 WAITTIMED

Tries to call an event from a semaphore, as long as no timeout occurs.

```
FUNCTION __CDECL VIRTUAL GLOBAL WAITTIMED
VAR_INPUT
    handle0      : MT_SEMAHANDLE;
    timeout0    : UDINT;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description

handle0	MT_SEMAHANDLE	Handle of the semaphore containing the event
timeout0	UDINT	The maximum waiting time (in milliseconds) until an event occurs
Return parameters	Type	Description
ret0	DINT	TRUE, if an event has been called. Otherwise no event was available and timeout has elapsed



If semahandle0 is an occupied resource or mutex semaphore, the (execution) priority of the tasks that occupy the semaphore is raised to that of the blocking task, if it has a higher priority (priority inheritance).

After the WAITTIMED() function has been completed processed, the active task occupies / possesses the semaphore. It cannot be suspended or terminated until it has released all of its resources.

Event semaphores are an exception. After successfully finishing the operation (ret0 = TRUE), they decrement the value of the semaphore with [WAIT\(\)](#).

2.2.4 Mailbox Functions

2.2.4.1 CLEARMAILBOX

Deletes the content of a mailbox.

```
FUNCTION __cdecl virtual global CLEARMAILBOX
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
mbhandle0	MT_MAILBOX	Handle of the mailbox to empty
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code



All data stored in the mailbox is discarded. If the mailbox is full and a task is waiting to write to the mailbox, CLEARMAILBOX() resets the task to ready (MT_Ready).

2.2.4.2 CREATEMAILBOX

Creates and initializes a mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL CREATEMAILBOX
VAR_INPUT
    messagelen0      : UDINT;
    messageslots0   : UDINT;
    name0           : ^char;
END_VAR
VAR_OUTPUT
    ret0            : MT_MAILBOX;
END_VAR;
```

Transfer parameters	Type	Description
messagelen0	UDINT	Length of a mailbox message in bytes
messageslots0	UDINT	Maximum number of messages the mailbox can save
name0	^char	Points to the name of the mailbox
Return parameters	Type	Description
ret0	MT_MAILBOX	Reference to the new mailbox or NIL, if an error occurred



LASAL OS inserts USR_ at the beginning of the string name. CREATEMAILBOX() reserves and initializes the mailbox.

2.2.4.3 DELETEMAILBOX

Releases the memory of a mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL DELETEMAILBOX
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
mbhandle0	MT_MAILBOX	Handle of the mailbox to delete
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code



LASAL OS checks, whether a task waits for a mailbox. In this case the program is canceled. The application has to care, that deleted mailboxes are no longer used.

2.2.4.4 GET

Calls a message from a mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL GET
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
mbhandle0	MT_MAILBOX	Handle of the mailbox containing the message to call
data0	pVoid	Points to the variable saving the message
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code



If the mailbox is empty, the calling task is blocked, until another task (or interrupt handler) saves a message. If the mailbox is full and another task is waiting to store a message in the mailbox, the task waiting is set to ready. If it has a higher priority, it is activated immediately.

2.2.4.5 GETCOND

Calls a message from a mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTFRONTCOND
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters		Type	Description
mbhandle0		MT_MAILBOX	Handle of the mailbox containing the message to call
data0		pVoid	Points to the variable saving the message
Return parameters		Type	Description
ret0		DINT	TRUE, if the message has been retrieved successfully, otherwise an error code is returned



Calls a message from the mailbox, if it is immediately available. GETCOND() never leads to a blocking task change. If the mailbox is empty, this is indicated by the return value and no data are transferred.

2.2.4.6 GETTIMED

Calls a message from a mailbox. If the mailbox is empty, there is a certain waiting time until a message is available again.

```
FUNCTION __CDECL VIRTUAL GLOBAL GETTIMED
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
    timeout0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters		Type	Description
mbhandle0		MT_MAILBOX	Handle of the mailbox containing the message to call
data0		pVoid	Points to the variable saving the message
timeout0		UDINT	Waiting time (in milliseconds) until a message is available in the mailbox
Return parameters		Type	Description
ret0		DINT	TRUE, if the message has been retrieved successfully, otherwise an error code is returned

2.2.4.7 MESSAGES

Returns the number of the currently saved messages in a mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL MESSAGES
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
END_VAR
VAR_OUTPUT
    ret0          : UDINT;
END_VAR;
```

Transfer parameters	Type	Description
mbhandle0	MT_MAILBOX	Handle of the mailbox to request
Return parameters	Type	Description
ret0	UDINT	Return value is always between 0 and the value of the messageslots0 parameter in CREATEMAILBOX() , as the mailbox was created

2.2.4.8 NEXTCOND

The next message is requested.

```
FUNCTION __CDECL VIRTUAL GLOBAL NEXTCOND
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
mbhandle0	MT_MAILBOX	Handle of the mailbox containing the requested message
data0	pVoid	Points to the variable saving the message
Return parameters	Type	Description
ret0	DINT	TRUE, if at least one message is available in the mailbox and data0 contains a copy of the message, otherwise an error code is returned



Requests the next message from a mailbox, if at least one message is available. Different to [GETCOND\(\)](#), the message is not called. NEXTCOND() never leads to a blocking task change.

2.2.4.9 PUT

Saves a message to a mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUT
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
mbhandle0	MT_MAILBOX	Handle of the mailbox where the message is saved
data0	pVoid	Is saved in the mailbox
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code



If the mailbox is full, the calling task is blocked, until another task (or interrupt handler) calls a message. If the mailbox is empty and another task is waiting for a message from the mailbox, the task waiting is set to ready. If it has a higher priority, it is activated immediately.

2.2.4.10 PUTCOND

Saves a message to a mailbox.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTCOND
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
mbhandle0	MT_MAILBOX	Handle of the mailbox where the message is saved
data0	pVoid	Is saved in the mailbox
Return parameters	Type	Description

ret0	DINT	TRUE, if the message has been saved successfully, otherwise an error code is returned
------	------	---------------------------------------------------------------------------------------



Saves a message to a mailbox, if enough space is available. PUTCOND() never leads to a blocking task change. If the mailbox is full, this is indicated by the return value and no data are transferred.

2.2.4.11 PUTFRONT

PUTFRONT() functions the same as [PUT\(\)](#), but adds the message at the beginning of the mailbox queue instead of at the end.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTFRONT
VAR_INPUT
    mbhandle0    : MT_MAILBOX;
    data0        : pVoid;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
mbhandle0	MT_MAILBOX	Handle of the mailbox where the message is saved
data0	pVoid	Is saved in the mailbox
Return parameters	Type	Description
ret0	DINT	MTERROR_NONE, if no error occurred, or an error code

2.2.4.12 PUTFRONTCOND

PUTFRONTCOND() functions the same as [PUTCOND\(\)](#), but adds the message at the beginning of the mailbox queue instead of at the end.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTFRONTCOND
VAR_INPUT
    mbhandle0    : MT_MAILBOX;
    data0        : pVoid;
END_VAR
VAR_OUTPUT
    ret0        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
---------------------	------	-------------

mbhandle0	MT_MAILBOX	Handle of the mailbox where the message is saved
data0	pVoid	Is saved in the mailbox
Return parameters	Type	Description
ret0	DINT	TRUE, if the message has been saved successfully, otherwise an error code is returned

2.2.4.13 PUTFRONTTIMED

PUTFRONTTIMED() functions the same as [PUTTIMED\(\)](#), but adds the message at the beginning of the mailbox queue instead of at the end.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTFRONTTIMED
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
    timeout0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
END_VAR;
```

Transfer parameters	Type	Description
mbhandle0	MT_MAILBOX	Handle of the mailbox where the message is saved
data0	pVoid	Is saved in the mailbox
timeout0	UDINT	Waiting time (in milliseconds) until space is available in the mailbox
Return parameters	Type	Description
ret0	DINT	TRUE, if the message has been saved successfully, otherwise an error code is returned

2.2.4.14 PUTTIMED

Saves a message in a mailbox. If the mailbox is full, there is a certain waiting time until space is available again.

```
FUNCTION __CDECL VIRTUAL GLOBAL PUTTIMED
VAR_INPUT
    mbhandle0      : MT_MAILBOX;
    data0          : pVoid;
    timeout0      : UDINT;
END_VAR
VAR_OUTPUT
    ret0          : DINT;
```

END_VAR;

Transfer parameters		Type	Description
mbhandle0		MT_MAILBOX	Handle of the mailbox where the message is saved
data0		pVoid	Is saved in the mailbox
timeout0		UDINT	Waiting time (in milliseconds) until enough space is available in the mailbox
Return parameters		Type	Description
ret0		DINT	TRUE, if the message has been saved successfully, otherwise an error code is returned

2.2.5 Interface Functions

If the OS interface class `_MultiTask` is not used, the header file `lsl_st_mt.h` linked so that the following functions can be used.

Task Functions

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		

Semaphore Functions

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 Functions

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		

Extended Functions

OS_MT_GetLastError	OS_MT_GetTime		
--------------------	---------------	--	--

All functions have the same parameters and return values as the functions _MultiTask class.

2.2.5.1 Examples

Creating a Task

Create a task from 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
```

Sending a Message

A task sends data directly to a second 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);

...
```

Semaphore Handling

Task 1 should be activated, when task 2 reaches a certain point.

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 Handling

Task 1 gets data from 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 Overview

The application heap is reserved for the application's memory requirements. The application uses the OS_SSR_Alloc, OS_SSR_ReAlloc and OS_SSR_Free functions to allocate, reallocate and de-allocate heap memory.

The size of the heap depends on the platform. The variables _UserHeapTotalSize (*AT % M 0024*), _UserHeapUsedMem (*AT % M 0028*) and UserHeapFreeMem (*AT % M 002C*), defined in the Rtos_variable.h header file can be used to monitor the total size as well as the amount of allocated and free space.

2.3.2 Using the Debug Heap

Some problems that programmers encounter are overwriting the end of an allocated buffer, writing before the allocated buffer and leaking memory, which is caused by failing to free allocations after they are no longer needed. The debug heap provides tools to solve these kinds of memory allocation problems.

When the debug heap is activated, the heap manager allocates a slightly larger block of memory than requested and returns a pointer to the portion of that block. The additional memory allocated by the debug heap routines is used for bookkeeping information, for pointers that link debug memory blocks together and for small buffers on either side of the data to catch overwrites of the allocated region.

The debug heap functions can be activated with the following CLI commands:

SET DBGHEAP

Display the current DBGHEAP settings.

SET DBGHEAP CHECK_ALWAYS ON|OFF

ON: The entire application heap is checked for integrity on every heap function call.

OFF: Only the block specified in OS_SSR_ReAlloc or OS_SSR_Free is checked according to the other DBGHEAP settings.

SET DBGHEAP ERR_OUTOFMEM ON|OFF

ON: The OS raises an AppMem error when OS_SSR_Alloc or OS_SSR_ReAlloc fails because of a memory shortage.

SET DBGHEAP REALLOC_NEW_PTR ON|OFF

ON: The OS_SSR_ReAlloc function always allocates a new block at a different memory location than the original block.

SET DBGHEAP CHECK_LIMIT_BEGIN ON|OFF

ON: The heap manager allocates a larger block than requested for a small buffer on the beginning side of your data to catch overwrites of the allocated region before the start of the region. The heap manager fills this buffer with 0xFD.

SET DBGHEAP CHECK_LIMIT_END ON|OFF

ON: The heap manager allocates a larger block than requested for a small buffer on the ending side of your data to catch overwrites of the allocated region after the end of the region. The heap manager fills this buffer with 0xFD.

SET DBGHEAP FILL_FREE_BLOCK ON|OFF

ON: The heap manager fills freed blocks with a known value (0xDD) to check that freed memory is not still being written to.

SET DBGHEAP FILL_NEW_BLOCK ON|OFF

ON: New blocks are filled with 0xCD when they are allocated.

SET DBGHEAP DBGHEAP_STORE_IP ON|OFF

ON: The heap manager allocates a larger block than requested for a buffer that holds the IP (instruction pointer) address where the heap function was called. In the case of an AppMem error the IP address of the error block is written to the system log.

SET DBGHEAP CHECK_FREE_PTR ON|OFF

ON: The OS raises an AppMem error when OS_SSR_Free specifies a memory block that was already freed.

SET DBGHEAP ALL ON|OFF

ON: Activate all of the above debug heap options.

OFF: Deactivate all of the above debug heap options

Note that activating any of the debug heap functions may degrade the performance of your application. The Debug-Heap functions are available since LasalOS 5.52.

2.3.3 SSR Interface Functions for the Heap

Header files: Isl_st_ifssr.h

2.3.3.1 OS_SSR_Malloc

The OS_SSR_Malloc() function allocates a memory block.

```
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)
```

Transfer parameters	Type	Description
size0	UDINT	Bytes to allocate
Return parameters	Type	Description
ret0	PVOID	OS_SSR_Malloc() returns a void pointer to the allocated space, or it returns NIL if there is insufficient memory available. To return a pointer to a type other than void, use a type cast on the return value.

The OS_SSR_Malloc() function allocates a memory block of at least size0 bytes. The block may be larger than size0 bytes because of space required for alignment and maintenance information.

2.3.3.2 OS_SSR_ReAlloc

The OS_SSR_ReAlloc() function reallocates a memory block.

```
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)
```

Transfer parameters	Type	Description
ptr0	PVOID	Pointer to a previously allocated memory block
size0	UDINT	New size in bytes
Return parameters	Type	Description

ret0	PVOID	OS_SSR_ReAlloc() returns a void pointer to the reallocated (and possibly moved) memory block. The return value is NIL if the size is zero and the buffer argument is not NIL, or if there is not enough available memory to expand the block to the given size. In the first case, the original block is freed. In the second case, the original block is unchanged. To get a pointer to a type other than void, use a type cast on the return value.
------	-------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The OS_SSR_ReAlloc() function changes the size of an allocated memory block. The ptr0 argument points to the beginning of the memory block. If ptr0 is NIL, OS_SSR_ReAlloc() behaves the same way as OS_SSR_Alloc and allocates a new block of size0 bytes. If ptr0 is not NIL, it should be a pointer returned by a previous call to OS_SSR_Alloc or OS_SSR_ReAlloc().

The size0 argument gives the new size of the block in bytes. The contents of the block are unchanged up to the shortest of the new and old sizes, although the new block can be in a different location.

The pointer returned by OS_SSR_ReAlloc() is not guaranteed to be the pointer passed through the ptr0 argument, because the new block can be in a new memory location.

2.3.3.3 OS_SSR_Free

The OS_SSR_Free() function deallocates or frees a memory block.

```
FUNCTION GLOBAL __cdecl P_SSR_Free
VAR_INPUT
    ptr0      : PVOID;
END_VAR;
#define OS_SSR_Fr
ee(pl) _LSL_POS^.piSSR^.SSR_Free $ P_SSR_Free(pl)
```

Transfer parameters	Type	Description
ptr0	PVOID	Previously allocated memory block to be freed

2.4 API Serial Interface

2.4.1 Overview

The Serial-User-API offers interrupt-driven communication through serial ports. Received data is buffered up to a defined limit before data is lost. To communicate over a serial port, the following steps should be taken:

- Open the port with SERUSER_Init.

- Applications that want to receive data should call SERUSER_RecvChar or SERUSER_RecvBlock.
- Applications that want to send data should call SERUSER_Send.
- When communication is complete, close the port with SERUSER_Close and optionally re-enable LASAL-online-communication with SERUSER_SetOnline.

2.4.2 Interface Functions SERIAL

Header files: lsl_st_serial.h

2.4.2.1 SERUSER_ClearRecvBuffer

The SERUSER_ClearRecvBuffer() function clears the receive buffer.

```
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)
```

Transfer parameters	Type	Description	
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init	
Return parameters	Type	Description	
ret0	DINT	0	Function successful
		<0	Negative error code

2.4.2.2 SERUSER_Close

The SERUSER_Close() function closes the serial interface.

```
FUNCTION GLOBAL __cdecl P_SerUsr_Close
VAR_INPUT
    handle0 : pVoid;
END_VAR;

#define SERUSER_Close(p1)
```

```
_LSL_POS^.piSerial^.pClose
$ P_SerUsr_Close(p1)
```

Transfer parameters	Type	Description
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()

This function does not re-enable the LASAL online communication.

2.4.2.3 SERUSER_EnableFIFO

The SERUSER_EnableFIFO() function enables or disables the operation of the receive and transmit 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)
```

Transfer parameters	Type	Description				
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()				
Trigger0	UDINT	Specifies the triggering level on the receive FIFO. When the number of bytes in the receive FIFO reaches the trigger level, a Received Data Available interrupt is set. A value of 0 disables the FIFO. To enable the FIFO, one of the following triggering levels can be specified: 1, 4, 8, and 14.				
Return parameters	Type	Description				
ret0	DINT	<table border="1"> <tr> <td>0</td><td>Success</td></tr> <tr> <td><0</td><td>Negative error code</td></tr> </table>	0	Success	<0	Negative error code
0	Success					
<0	Negative error code					

When you disable the transmit and receive FIFOs, you will lose all data stored in the FIFO buffers. When you call this function to enable the FIFO but the UART has no FIFO, then the function fails.

Requirements

LasalOS: Requires LasalOS 5.28 or later.

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)
```

Transfer parameters	Type	Description	
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()	
OnOff0	^UDINT	A variable that receives the current value of the RS422/485 output driver. A value of zero indicates that the output driver is not selected.	
Return parameters	Type	Description	
	DINT	0 Function successful <0 Negative error code	

Requirements

LasalOS: Requires LasalOS 5.28 or later.

2.4.2.5 SERUSER_GetError

The SERUSER_GetError() function retrieves the serial interface's last-error code value.

```
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)
```

Transfer parameters	Type	Description
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()
Return parameters	Type	Description
ret0	DINT	The return value is the calling serial interface's last error code value. This value can be one of the list below.
Code	Name	Description
0	SERERROR_NONE	No error.
-1	SERERROR_COMM	Either an invalid number of the serial interface was specified or the function is not supported for this interface.
-2	SERERROR_BAUDTABLE	Error baud table
-3	SERERROR_BAUDRATE	The baudrate parameter is incorrect.
-4	SERERROR_PARITY	The parity parameter is incorrect.
-5	SERERROR_STOPBIT	The stop bit parameter is incorrect.
-6	SERERROR_WORDLEN	The wordlen parameter is incorrect.
-10	SERERROR_INUSE	The specified serial interface is already in use by the application.

-11	SERERROR_OSINUSE	COM interface is already used by the OS.
-12	SERERROR_NOTAVAILABLE	The specified serial interface is not available on this system.
-13	SERERROR_NOMEM	There is not enough storage available to process this command.
-14	SERERROR_NOHANDLE	The handle is invalid.
-15	SERERROR_PARAMETER	The parameter is incorrect.
-16	SERERROR_RECVBUF	The receive buffer parameters are incorrect.
-17	SERERROR_SENDBUF	The send buffer parameters are incorrect.
-19	SERERROR_RECVERROR	The SERUSER_RecvChar or SERUSER_RecvBlock functions were called but there is no data in the receive buffer.
-20	SERERROR_SENDERROR	The send process could not be completed (interrupt, send buffer full).
-21	SERERROR_GENERAL	Internal error during initialization or operation of the serial interface.
-22	SERERROR_PARITY_E	Received data packets were discarded due to an invalid parity.
-23	SERERROR_FRAMING_E	Received data packets were discarded due to framing errors (i.e. wrong baud rate, word length etc.).

2.4.2.6 SERUSER_GetInfo

The SERUSER_GetInfo() function retrieves the current control settings and used resources for a specified serial interface.

```

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)

```

Transfer parameters		Type	Description	
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()		
info0	^LSLAPI_SERIALINFO	Pointer to a LSLAPI_SERIALINFO structure that receives the information		
Return parameters		Type	Description	
ret0	DINT	0 Success <0 Error		

LSLAPI_SERIALINFO Structure

```
LSLAPI_SERIALINFO : STRUCT
  initialized      : USINT;
  comportnum       : USINT;
  IRQNum          : USINT;
  IOPort           : UINT;
  Baudrate         : UINT;
  Ptr_RecvBuffer   : pVoid;
  Ptr_SendBuffer   : pVoid;
END_STRUCT;
```

Elements of the LSLAPI_SERIALINFO structure.

Initialized	Indicates whether the serial interface is initialized
Comportnum	This is a zero-based index of the serial interface
	 Note that common-parameters in SERUSER functions are ONE-based!
IRQNum	IRQ number of the serial interface
IOPort	IO base address of the serial interface
Baudrate	Baud rate at which the serial interface operates
Ptr_RecvBuffer	Pointer to the receive buffer
Ptr_SendBuffer	Pointer to the send buffer

Remarks

This function should only be used for informational purposes, e.g. it is not recommended, that the registers of the serial interface are accessed directly!

2.4.2.7 SERUSER_GetModemControl

The SERUSER_GetModemControl() function queries the modem control register (MCR).

```
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
```

Transfer parameters	Type	Description	
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()	
Value0	^USINT	Pointer to a USINT that receives the content of the modem control register (MCR)	
Return parameters	Type	Description	
	DINT	0 SERUSER_Init <0 Negative error code	

Requirements

LasalOS: Requires LasalOS 5.28 or later.

2.4.2.8 SERUSER_GetModemStatus

The SERUSER_GetModemStatus() function queries the modem status register (MSR).

```
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)

```

Transfer parameters		Type	Description	
handle0		pVoid	Handle to the serial interface returned by SERUSER_Init()	
Value0		^USINT	Pointer to a USINT that receives the content of the modem status register (MSR)	
Return parameters		Type	Description	
ret0		DINT	0 Function successful <0 Negative error code	

Requirements

LasalOS: Requires LasalOS 5.28 or later.

2.4.2.9 SERUSER_GetRecvStatus

The SERUSER_GetRecvStatus() function returns the number of characters in the receive buffer.

```

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)

```

Transfer parameters		Type	Description	
handle0		pVoid	Handle to the serial interface returned by SERUSER_Init()	
Return parameters		Type	Description	

ret0	DINT	>0 Number of characters currently in the receive buffer <0 Negative error code
------	------	-----------------------------------------------------------------------------------

2.4.2.10 SERUSER_GetSendStatus

The SERUSER_GetSendStatus() function returns the number of characters in the send buffer.

```
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)
```

Transfer parameters	Type	Description
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()
Return parameters	Type	Description
ret0	DINT	>0 Number of characters in the send buffer <0 Negative error code

2.4.2.11 SERUSER_Init

The SERUSER_Init() function opens a serial interface, configures it and returns a handle that can be used to access the interface. When the interface is also used by the LASAL online communication, then the LASAL online communication is disabled. The LASAL online communication is re-enabled when the project stops or when a call to [SERUSER_SetOnline\(\)](#) enables it.

```
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)

```

Transfer parameters	Type	Description																						
commnum0	UINT	<p>Specifies the number of the serial interface</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>Specifies the baud rate at which the serial interface operates</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>Specifies the parity scheme to be used</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> <tr><td>3</td><td>SERUSERPARITY_MARK</td></tr> </table>	0	SERUSERPARITY_NONE	1	SERUSERPARITY_ODD	2	SERUSERPARITY_EVEN	3	SERUSERPARITY_MARK														
0	SERUSERPARITY_NONE																							
1	SERUSERPARITY_ODD																							
2	SERUSERPARITY_EVEN																							
3	SERUSERPARITY_MARK																							

		4 SERUSERPARITY_SPACE
stopbits0	UINT	<p>Specifies the number of stop bits to be used</p> <p>1 One stop bit</p> <p>2 2 stop bits for words of length of 6, 7 or 8 bits or 1.5 stop bits for word lengths of 5 bits</p>
wordlength0	UINT	<p>Specifies the number of bits in the bytes transmitted and received. Possible values are 5, 6, 7, 8. A word length of 8 bits is most commonly used today.</p>
Return parameters	Type	Description
ret0	pVoid	<p>If the function succeeds, the return value is an open handle to the specified serial interface. If the function fails, the return value is a NIL pointer. To get extended error information, call SERUSER_GetError().</p>

Use [SERUSER_Close\(\)](#) to close the serial interface.

2.4.2.12 SERUSER_RecvBlock

The SERUSER_RecvBlock() function reads data from the serial drivers receive buffer.

```
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)
```

Transfer parameters	Type	Description
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()
buffer0	pVoid	Pointer to the buffer that receives the data read from the receive buffer
rdlength0	UDINT	Specifies the number of bytes to read

rdlen0	^UDINT	Pointer to an UDINT to receive the number of bytes actually reading. This value may be less than rdlength0.
Return parameters	Type	Description
ret0	DINT	0 Function successful, at least one byte was available <0 Negative error code

2.4.2.13 SERUSER_RecvChar

The SERUSER_RecvChar() function reads one byte from the serial drivers receive buffer.

```
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)
```

Transfer parameters		Type	Description
handle0		pVoid	Handle to the serial interface returned by SERUSER_Init()
buffer0		pVoid	Pointer to a character that receives the byte read from the receive buffer
Return parameters		Type	Description
ret0		DINT	0 Function successful, a byte was available <0 Negative error code

2.4.2.14 SERUSER_Send

The SERUSER_Send() function sends data asynchronously using interrupts. The data is placed in a send buffer to be transmitted by the interrupt handler as soon as the transmit register becomes empty.

```
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)
    _LSL_POS^.piSerial^.pSend
    $ P_SerUsr_Send(p1,p2,p3,p4)

```

Transfer parameters	Type	Description	
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()	
buffer0	pVoid	Pointer to a buffer containing the data to be sent	
bufferlength0	UDINT	Length of the data buffer to be sent	
wrlen0	^UDINT	Pointer to an UDINT to receive the number of bytes placed in the send buffer. Usually, wrlen0^ will contain bufferlength0 after the call. However, in case of send buffer shortage, the returned value may be less. wrlen0 may be set to NIL if this information is not required by an application.	
Return parameters	Type	Description	
ret0	DINT	0 Function successful <0 Error	

2.4.2.15 SERUSER_Set422Mode

The SERUSER_Set422Mode() function controls the output driver of a RS422/RS485 interface.

```

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)
```

Transfer parameters		Type	Description	
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()		
recvbuffer0	UDINT	0	Output driver is switched off	
		1	Output driver is switched on	
Return parameters		Type	Description	
ret0	DINT	0	Function successful	
		<0	Error	

The RS422 and RS485 mode is not supported on all platforms. Please refer to the documentation of your hardware. Since RS422 there is a point-to-point connection, the output driver can be continuously on in this mode. In RS485, the output driver can be on only when data is sent, because RS485 uses a multimaster connection type.

Requirements

LasalOS: Requires LasalOS 5.28 or later.

2.4.2.16 SERUSER_SetBufferRecv

This function specifies a new receive buffer.

```
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)
  _LSL_POS^.piSerial^.pSetBufferRecv
  $ P_SerUsr_SetBufferRecv(p1,p2,p3)
```

Transfer parameters		Type	Description	
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()		

recvbuffer0	pVoid	Pointer to a receive buffer				
bufferlength0	UDINT	Size of recvbuffer0 in bytes; this value must be at least 128 bytes				
Return parameters	Type	Description				
ret0	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: center;">0</td><td>Function successful</td></tr> <tr> <td style="text-align: center;"><0</td><td>Error</td></tr> </table>	0	Function successful	<0	Error
0	Function successful					
<0	Error					

2.4.2.17 SERUSER_SetModemControl

The SERUSER_SetModemControl() function sets or clears individual bits of the modem control register (MCR) of the UART. For a description of this register please refer to a UART documentation.

```
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)
    _LSL_POS^.piSerial^.pSetModemControl
    $ P_SerUsr_SetModemControl(p1,p2,p3)
```

Transfer parameters	Type	Description		
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()		
SetToOneZero0	UDINT	This parameter specifies which operation should be performed on the modem control register. A value of 0 performs a bitwise logical OR with MCR and NewValue0. A value of 1 performs a bitwise logical NOT operation on NewValue0 and then a bitwise logical AND operation with MCR and the negated NewValue0. The result of this operation is written to the modem control register. With SetToOneZero0 = 0 you can set individual bits, and with SetToOneZero0 = 1 you can clear individual bits of the MCR.		
NewValue0	UDINT	Contains the value that is used to calculate the new value of the MCR. See description of parameter SetToOneZero0.		
Return parameters	Type	Description		
ret0	DINT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; text-align: center;">0</td><td>Function successful</td></tr> </table>	0	Function successful
0	Function successful			

		<0 Negative error code
--	--	------------------------

Requirements

LasalOS: Requires LasalOS 5.28 or later.

2.4.2.18 SERUSER_SetOnline

The SERUSER_SetOnline() function enables or disables the LASAL online communication. When the LASAL online communication was disabled by the function [SERUSER_Init\(\)](#), then use this function to re-enable LASAL online communication.

```
FUNCTION GLOBAL __cdecl P_SerUsr_SetOnline
VAR_INPUT
    action0 : UDINT;
END_VAR
VAR_OUTPUT
    ret0 : DINT;
END_VAR;

#define SERUSER_SetOnline(p1)
    _LSL_POS^.piSerial^.pSetOnline
    $ P_SerUsr_SetOnline(p1)
```

Transfer parameters		Type	Description
action0	UDINT	Specifies whether the LASAL online communication should be enabled.	
0 Disabled			
1 Enabled			
Return parameters		Type	Description
ret0	DINT	0 Function successful	
<0 Error			

Do not call this function when the serial interface is open!



2.4.2.19 SERUSER_UserFunction

The SERUSER_UserFunction() function is used to install a high-level user interrupt handler function. When an interrupt on the serial port is triggered, the low-level handler in the operating system calls the high-level handler. The operating system does not clear the interrupt in the UART. It is the responsibility of the user function to determine the reason of the interrupt and to read the appropriate registers to clear the interrupt in the UART.

```
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)
```

Transfer parameters	Type	Description
handle0	pVoid	Handle to the serial interface returned by SERUSER_Init()
function0	pVoid	Pointer to the user interrupt function. The type of this function is as follows: FUNCTION GLOBAL __cdecl P_SerUsr_UsrIntFcn VAR_INPUT userParam : pVoid; comNum : pVoid; ioPort : pVoid; END_VAR
UserParam		Pointer that is specified in parameter param0 of this function
comNum		It is a zero-based index of the serial interface.  Note that common-parameters in SERUSER functions are one-based!
ioPort		IO base address of the serial interface
param0	pVoid	Contains a pointer to be passed to the user interrupt function
Return parameters	Type	Description

ret0	DINT	0 Function successful <0 Error
------	------	-----------------------------------

Requirements

LasalOS: Requires LasalOS 5.28 or later.

2.5 API Serial Number

2.5.1 General

This interface provides the serial PLC and IDE drive numbers.

2.5.2 Structures Used in ISYSSERNUM

Header files: lsl_st_syssernum.h

2.5.2.1 tagPLCInfo

Structure containing PLC Info.

```
// 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
```

Parameters

udVersion	Version number of this structure. Future releases may have more components.
szBIOSVersion	The BIOS version number of the PLC, if available.
szSerialVersion	The serial number of the PLC.
szApplication	The name of the current application.

These components are currently available for IPCs and C-IPCs.

Requirements

Version: Lasal OS V5.51 or higher.

2.5.3 Interface Functions ISYSSERNUM

Header files: lsl_st_syssernum.h

This section describes the interface connection for the serial numbers. To use this interface, a pointer must first be obtained via "OS_CILGET". The name of the interface is "ISYSSERNUM". If the same name for the pointer is used as in the following example, the predefined macros can be used to reduce and simplify coding.

The serial numbers are returned as non-zero terminated ASCII-strings!

2.5.3.1 SernumGetPLC

Retrieves the serial number of the PLC.

```
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)
```

Transfer parameters		Type	Description	
perNum		^Void	Pointer to the buffer, which will receive the serial number	
ulBufLen		UDINT	Size of buffer	
Return parameters		Type	Description	
retval		DINT	≥ 0 Number of characters copied into the buffer < 0 Error	

Example

```
#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

```

Requirements

Version: LasalOS 5.43 or later.

2.5.3.2 SernumGetPLCDrive

Retrieves the serial number of PLC IDE and USB drives, if available.

```

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)

```

Transfer parameters	Type	Description	
ucDrive	USINT	Drive letter of the desired drive; ranges are from 'A' to 'Z' and from 'a' to 'z' are valid	
pSerNum	^Void	Pointer to the buffer, which will receive the serial number	
ulBufLen	UDINT	Size of the buffer	
Return parameters	Type	Description	
retval	DINT	≥ 0 Number of characters copied into the buffer <0 Error	

Example

see above

Requirements

Version: LasalOS 5.43 or later.

2.5.3.3 SernumGetPLCInfo

Retrieves some general information about the PLC.

```
PPLCINFO SernumGetPLCInfo(  
void  
>;  
FUNCTION __CDECL GLOBAL P_SernumGetPLCInfo  
VAR_INPUT  
END_VAR  
VAR_OUTPUT  
    retval : ^LSL_PLCINFO;  
END_VAR;  
#define ISYSSENUM_SERNUMGETPLCINFO()  
    pISysSernum^.SernumGetPLCInfo  
    $ P_SernumGetPLCInfo()
```

Return parameters	Type	Description
retval	^LSL_PLCINFO	Pointer to a PLC INFO structure

Example

see above

Requirements

Version: LasalOS 5.51 or later.

2.6 API Sysmsg Interface

2.6.1 Overview

The Sysmsg API functions enable the application to produce error, warning and other messages. It is important that these messages can be written to a file to be viewed later.

A message is not written to a file immediately when it is generated, instead, it is first written to a buffer. This buffer is written to a file when the application stops, is reset, an exception occurs or when a power down is detected. In addition, the messages can be written to a file by calling a certain Sysmsg API function.

Since the message buffer is limited in size, messages can be lost when the buffer is filled before its content is written to a file. There is no tool that allows writing the buffer to a file automatically, as continually writing data to a disk is dangerous and decreases the life span of the disk. It is important the application programmer keep this in mind when writing messages to a file. In the event of a buffer overflow, the operating system records the time it occurred. An additional entry is then generated the next time the buffer is written to a file that indicates an overflow has occurred.

A limit, called a file quota, can be defined for the message file to avoid filling the disk completely. If the message size exceeds the file quota, the message is reassigned as a backup file and a new message file is created. When this occurs, however, an existing backup file is lost. The disk size therefore required for message files is $2 * \text{file quota}$; the file name is EVENTxx.LOG, where xx represents a unique number for every message buffer object. The path to where the message file is stored can be configured using the APPMSGPATH environment variable. The default path is C:\SYSMSG.

If only last few messages are important, a buffer overflow can be accepted. The application does not write the content of the message buffer to a file, this is performed by the operating system when the application ends. Otherwise the application has to monitor the amount of space used in the message buffer and eventually write the content of the buffer to a file.

To generate messages from an application, follow the steps below:

- Create a message buffer object with [OS_SYSMSG_LCREATE\(\)](#).
- When the message buffer object should be used in different parts of the application program, additional handles for the previously created object can be requested with a call to [OS_SYSMSG_LOPEN\(\)](#).
- Write data or text to the message buffer with either the OS_SYSMSG_LWRITE_x, OS_SYSMSG_LPRINTFLN_x, or OS_SYSMSG_LWRITE_I_x function; x is a placeholder for different expressions of these functions.
- The amount of used space in the buffer can also be monitored with [OS_SYSMSG_LINFO\(\)](#) and the content written to a file with [OS_SYSMSG_LFLUSH\(\)](#).
- The [OS_SYSMSG_LCLOSE\(\)](#) function can be used to close message buffer handles when the objects are no longer needed, although this is done by the operating system when the application ends.

The Sysmsg API functions are thread-save, which means that they can be called in any task and must not be sequential. Except for OS_SYSMSG_LWRITE_I0-4, these functions cannot be called from an interrupt.

2.6.2 Interface Functions SYSMSG

Header files: lsl_st_sysmsg.h

2.6.2.1 OS_SYSMSG_LCLOSE

The OS_SYSMSG_LCLOSE() function closes an open handle to a message buffer object.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LCLOSE
VAR_INPUT
    hLog      : UDINT;
END_VAR;

#define OS_SYSMSG_LCLOSE(p1)
    OS_pLslSysMsg^.lclose $ P_LSLSYSMSG_LCLOSE(p1)
```

Transfer parameters	Type	Description
hLog	UDINT	Handle to the message buffer object

If all open handles for a message buffer object are released with a call to OS_SYSMSG_LCLOSE(), the content can be written to a message file. The function writes the content of the message buffer to a file. When the application ends, the operating system closes all open handles to message buffer objects.

2.6.2.2 OS_SYSMSG_LCREATE

The OS_SYSMSG_LCREATE() function creates a new message buffer object and returns a handle, which can be used to access the object.

```
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)
```

Transfer parameters	Type	Description

nID	DINT	Specifies the identification number of the message buffer object. Its value must be between 0 and 9. This number is used to construct the file name of the message file. When more than one message buffers are used, each buffer must have its own identification number. The following IDs are reserved for SIGMATEK software components: 9,8,7 (Loader), 6 (Software packet ,Industry 4.0').
buffer	^CHAR	Pointer to a buffer where the messages are written to
bufferSize	UDINT	Size of the buffer
file_quota	DINT	Specifies the file-quota of the message file. A value of 0 disables writing the message buffer to a message file. A value of -1 indicates that the system default value should be used.
flags	UDINT	This parameter is reserved for future use and must be set to 0
Return parameters	Type	Description
hLog	UDINT	If the function succeeds, the return value is a handle to the message buffer. Otherwise the return value is 0.

A file-quota of 0 only makes sense for operating system messages when a static RAM is used for the message buffer or when a debug tool is available to load the messages up into the buffer.

2.6.2.3 OS_SYSMSG_LFLUSH

Flushes a log buffer to a log file on disk. The data is appended to the end of the log file. When logging to disk is not enabled, the function returns with an error.

```
FUNCTION GLOBAL __cdecl P_LSLSYSMSG_LFLUSH
VAR_INPUT
    hLog          : UDINT;
END_VAR
VAR_OUTPUT
    result        : DINT;
END_VAR;

#define OS_SYSMSG_LFLUSH(p1)
    OS_pLslSysMsg^.lflush
    $ P_LSLSYSMSG_LFLUSH(p1)
```

Transfer parameters	Type	Description
hLog	UDINT	Handle to the message buffer object
Return parameters	Type	Description

result	DINT	0 Buffer was successfully flushed, or specified log buffer has no data -1 Error
--------	------	------------------------------------------------------------------------------------

2.6.2.4 OS_SYSMSG_LINFO

Get information about a log buffer.

```
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)
```

Transfer parameters		Type	Description
hLog		UDINT	Handle to the message buffer object
buf_addr		^UDINT	Pointer to a variable where the address of the message buffer is stored (optional)
pos		^UDINT	Pointer to a variable where the current position in the message buffer is stored (optional)
bufsize		^UDINT	Pointer to a variable where the size of the message buffer without the preceding control structure is stored (optional)
n_unflushed		^UDINT	Pointer to a variable where the number of bytes that are not written to the message buffer is stored (optional)
Return parameters		Type	Description
result		UDINT	

2.6.2.5 OS_SYSMSG_LOPEN

The OS_SYSMSG_LOPEN() function opens an existing message buffer object and returns a handle that can be used to access the object.

```
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)
```

Transfer parameters	Type	Description
nID	DINT	Specifies the identification number of the message buffer object. Its value must be between 0 and 9.
Return parameters	Type	Description
hLog	UDINT	If the function succeeds, the return value is a handle to the message buffer. Otherwise the return value is 0.

2.6.2.6 OS_SYSMSG_LPRINTFLN1-4

Print formatted data to a log buffer. A new-line character is appended to the end of the text.

```
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 ,0 )
```

```
#define OS_SYSMSG_LPRINTFLN3(p1,p2,p3,p4,p5,p6)
OS_pLs1SysMsg^.lprintfln
$ P_LSLSYSMSG_LPRINTFLN(p1,p2,p3,p4,p5,p6,0 )
#define OS_SYSMSG_LPRINTFLN4(p1,p2,p3,p4,p5,p6,p7)
OS_pLs1SysMsg^.lprintfln
$ P_LSLSYSMSG_LPRINTFLN(p1,p2,p3,p4,p5,p6,p7)
```

Transfer parameters		Type	Description
hLog	UDINT	Handle to the message buffer object	
fAddTimestamp	UDINT	A flag that specifies whether a timestamp should be added to the message. A value of 0 means that no timestamp is added, all other values add a timestamp.	
msg	^CHAR	0-terminated format control string	
lpar1-4	UDINT	Optional arguments	
Return parameters		Type	Description
result	UDINT	The return value is the number of characters written, not including the terminating null character, or a negative value if an output error occurs. If the number of characters to write exceeds 256, then 256 characters are written. In this case the message should be divided into more parts.	

OS_SYSMSG_LPRINTFLNx formats and prints a series of characters and values to the message buffer. Each function is converted and output according to the corresponding format specification in msg. The msg argument has the same syntax and use that it has in the print function of the C programming language.

2.6.2.7 OS_SYSMSG_LWRITE

The OS_SYSMSG_LWRITE() function writes data to a message buffer.

```
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_pLs1SysMsg^.lwrite
$ P_LSLSYSMSG_LWRITE(p1,p2,p3,p4)
```

Transfer parameters	Type	Description
hLog	UDINT	Handle to the message buffer object
fAddTimestamp	UDINT	Flag that specifies whether a timestamp should be added to the message or not. A value of 0 means that no timestamp is added, all other values add a timestamp.
data	^CHAR	Pointer to the data that should be written to the message buffer
size	UDINT	Size of data
Return parameters	Type	Description
result	UDINT	Number of bytes written to the message buffer

2.6.2.8 OS_SYSMSG_WRITE_I

Writes a string and up to four optional parameters of type UDINT to a log buffer.

```
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;
```

Transfer parameters	Type	Description
hlog	UDINT	Handle to the message buffer object
pTxt	^CHAR	0-terminated format control string. This format string must contain specifications that determine the output format for the arguments. For a description of the format string see the documentation of the printf function in C.
udParam1-4	UDINT	Optional arguments
Return parameters	Type	Description
result	UDINT	Returns the length of the string written to the log buffer

A new line character is attached to the end of the text; this function can be called from an interrupt. If the length of a string is greater than 31, it is truncated to a length of 31. When an optional parameter is used, the string must contain the appropriate print format

specification. These function parameters are first queued in an interrupt log buffer and then later printed to the actual log buffer. For this reason, caution must be taken to ensure that an optional parameter does not reference data that is only valid in the interrupt function. For example, a local Character array cannot be declared in the stack, and a format string of "%s".

The entries are removed from the interrupt log buffer as soon as any of the SYSMSG API functions are called. If this function is called too often the internal queue can fill up. In such an event, the oldest message is discarded. Each entry is assigned a number that is also printed in the log; this allows the user to see whether messages have been lost.

2.6.2.9 OS_SYSMSG_LWRITELN

Writes a 0-terminated string to a log-buffer with a new-line character at the end of the data.

```
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_pLslSysMsg^.lwriteLn
    $ P_LSLSYSMSG_LWRITELN(p1,p2,p3)
```

Transfer parameters		Type	Description
hLog		UDINT	Handle to the message buffer object
fAddTimestamp		UDINT	Flag that specifies whether a timestamp should be added to the message. A value of 0 means that no timestamp is added, all other values add a timestamp.
txt		^CHAR	Pointer to a 0-terminated string
Return parameters		Type	Description
result		UDINT	Number of bytes written to the message buffer

2.6.3 Sysmsg – Changes in Newer OS

There are a few changes in Lasal OS versions ≥ 5.80 (C-IPC 5.60 KM):

- New Timestamp.
- Log buffer will be flushed automatically.
- New file format.
- Extended Userlog.

Timestamp

The timestamp added to message contains milliseconds.

Logbuffer will be Flushed Automatically

If the space of the log buffer becomes very low, the buffer will be flushed. This feature is only available for the System Log Buffer (ID 0) and the extended Userlog buffers (described below) (ID 1, ID 2).

File Format

At the beginning of the log file there's a Sigmatek-specific header containing the log buffer ID and the OS versions. The header of the userlog files have an additional entry (see below). The entries of each line are separated by a semicolon (;).

Extended Userlog

There are two log buffers (ID 1 – low priority and ID 2 – high priority) allocated by the OS, so it is not necessary to create, open or close a log buffer. The log files are always saved as 'event01.log' for the low priority and 'event02.log' for the high priority log buffer. On an OS-Update from < 5.80 (C-IPC 5.60 KM) to ≥ 5.80 the old log file will be renamed from 'eventxx.log' to 'eventxx.err' because the OS doesn't finds valid file header. If no logfile exists, the OS creates a new one on start-up. The userlog files have an extended user file header after the system file header for user entries. The OS fills the user header with a default message. Each line of the file header is providing with a timestamp. The logfiles will be flushed when the application ends.

2.6.4 Extended Userlog Functions

Header files: lsl_st_sysmsg.h

2.6.4.1 OS_USERLOG_EXPORT

Copies all userlog files and the System Eventlog file (ID 0) with backup files to a given path.

```
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_pLslSysMsg^.plprintfln_userlog
    $ P_LSLSYSMSG_LUSER_EXPORT(p1)
```

Transfer parameters		Type	Description
path		^CHAR	The destination path
Return parameters		Type	Description
rc		DINT	0 successful <0 error code

The path must be a legal filename syntax ("Drive:\Directory", "\Directory\...", "Directory"). If path = NIL, the files will be copied to the current directory. If the files in the destination path already exist, they will be overwritten. The function will not block. It returns LUSER_BUSY until all files are copied successful or an error occurs. The return value only gives information of the successful call to the function and to get LUSERBUSY state. It gives no information if the files were copied successfully. [OS_USERLOG_LAST_EXPORT_RESULT\(\)](#) returns the result of the copy process.

2.6.4.2 OS_USERLOG_FILEHEADER

Writes a user file header to the log file.

```
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)

```

Transfer parameters	Type	Description	
str	^CHAR	The file header message as a null-terminated string	
LogFileID	DINT	The log file to write the header to	
bNonBlocking	USINT	0 Function blocks 1 Function will not block	
Return parameters	Type	Description	
rc	DINT	0 Function successful <0 Error code	

The longest file header message that can be written is 106 bytes. If the number of characters exceeds 106, 106 characters are written. The user file header is only available for log file ID 1 and ID 2. If the bNonBlocking parameter is 0, the function blocks as long as the function needs to write the file header. If the bNonBlocking parameter is 1, the function will not block and returns LUSER_BUSY until it is finished writing the header or an error occurred.

2.6.4.3 OS_USERLOG_FLUSH

The function flushes the given log buffer to disk.

```

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)

```

Transfer parameters		Type	Description	
LogFileID		DINT	LUSER_LOW_PRIO_ID or 1 for the low-priority log buffer LUSER_HIGH_PRIO_ID or 2 for the high-priority log buffer	
bNonBlocking		USINT	0 Function blocks 1 Function will not block	
Return parameters		Type	Description	
rc		DINT	0 Function successful <0 Error code	

The function flushes the log buffer to the path set by the environment variable APPMSGPATH or to the default system path C:\SYSMSG. If the bNonBlocking parameter is 0, the function blocks as long as the function needs to flush the log buffer. If the bNonBlocking parameter is 1, the function will not block and returns LUSER_BUSY until the flush process becomes ready.

2.6.4.4 OS_USERLOG_LAST_EXPORT_RESULT

The function is used to get information about the last [OS_USERLOG_EXPORT\(\)](#) call.

```
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_pLs1SysMsg^.plprintfln_userlog
    $ P_LSLSYSMSG_LUSER_LAST_EXPORT_RESULT(p1)
```

Transfer parameters		Type	Description	
PtrExpResult		^PLUSER_EXPORT_RESULT	Pointer to a pointer that points to an OS allocated array that stores the export result	
Return parameters		Type	Description	

errcount	DINT	≥0	Number of errors occurred in the copy process
		<0	Negative error code

Example

```

ExportResult      :^LUSER_EXPORT_RESULT;
OS_USERLOG_LAST_EXPORT_RESULT(#ExportResult);

TYPE
  LUSER_EXPORT_RESULT : STRUCT
    rc      : DINT; // error code from the file functions
    LogFileID      : DINT; // ID from the logfile
    IsBackupFile    : USINT; // 0 .. logfile (.log), 1 .. backupfile (.bak)
    pNext      :^LUSER_EXPORT_RESULT;
  END_STRUCT;
  PLUSER_EXPORT_RESULT  :^LUSER_EXPORT_RESULT;
END_TYPE

```

The array will be filled with all results, even if no error occurred. Use the pNext pointer to get the next entry until pNext = NIL. lsl_st_osfile describes the file error codes.

rc: -1000 .. dest = NIL internal error.
 -1001 .. Not enough memory to alloc an internal buffer to copy the files.

2.6.4.5 OS_USERLOG_PRINTFNLN (1-2)

Writes a message to the low-priority log buffer.

```

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)

```

Transfer parameters		Type	Description
msg		^CHAR	0-terminated format control string
param1-2		DINT	Optional arguments
Return parameters		Type	Description
rc		DINT	>0 Number of bytes written <0 Error code (see Isl_st_sysmsg.h)

The longest message that can be written is 234 bytes. If the number of characters exceeds 234, then 234 characters are written. If the space in the log buffer becomes low, the OS flushes the log buffer. The function will not block, it returns LUSER_BUSY during a flush is in process. In this case no messages will be logged until the flush process becomes ready.

2.6.4.6 OS_USERLOG_WRITEEVENT

The function writes 5 values to the high-priority log buffer.

```
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)
```

Transfer parameters		Type	Description
hID-Y			Values to write
Return parameters		Type	Description
rc		DINT	>0 Number of bytes written <0 Error code (see Isl_st_sysmsg.h)

If the space in the log buffer becomes low, the OS flushes the log buffer. The function will not block, it returns LUSER_BUSY, during a flush is in process. In this case no messages will be logged until the flush process becomes ready. This function can be called from REALTIME tasks.

2.7 API TCP User Interface

2.7.1 Error Values

Cannot return an error string

TCP_NOT_READY	-2000	Not ready. Examples: OS_TCP_USER_CONNECT(), OS_TCP_USER_ACCEPT(): no connection has been established yet. OS_TCP_USER_RECV(): no receive data is available. OS_TCP_USER_SEND(): the remote host is currently not ready to receive data.
TCP_INVALID_SOCKET_STATE	-2001	Socket is in the wrong state. Examples: When OS_TCP_USER_CONNECT() is called, the socket is not in the correct state. OS_TCP_USER_NREAD() or OS_TCP_USER_IOCTLSOCKET() or OS_TCP_USER_RECV() or OS_TCP_USER_SEND() etc. are called without having previously called OS_TCP_USER_ACCEPT() or OS_TCP_USER_CONNECT(). OS_TCP_USER_ACCEPT() without previous OS_TCP_USER_LISTEN().
TCP_NOMEM_ERROR	-2002	Too little free memory. Examples: OS_TCP_USER_SOCKET(): too many sockets were created. OS_TCP_USER_STRTOULONG_ASY: internal error.
TCP_BUFFER_TO_SMALL	-2003	The target buffer is too small. Examples: OS_TCP_USER_TOIP(), OS_TCP_USER ULONGTOSTR(), OS_TCP_USER_GETERRORSTRING(). OS_TCP_USER_STRTOULONG_ASY: the mailbox is full (internal error).

TCP_MAXSOCKETS_ERROR	-2004	All sockets are in use.
TCP_INVALID_SOCK_NUM	-2005	The transferred socket number is invalid.
TCP_INVALID_NUM_BER	-2007	Invalid input value. Examples: A component of a binary IP address is greater than 255: OS_TCP_USER_TOIP().
TCP_SYSTEM_ERROR	-2008	Error reading the socket info.
TCP_NOIF_ERROR	-2009	Number of the network interface invalid. Z.B. OS_TCP_USER_IPINFO().

Can return an error string

TCP_ADDR_NOT_AVAIL	-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_UNREACH	-3012	destination is unreachable (ICMP)
TCP_INVALID_PARAM	-3013	invalid parameter (pointer is 0, etc.)
TCP_IFACE_CLOSED	-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_CONNECTED	-3019	socket is already connected
TCP_MC_TABLE_FULL	-3020	multicast table full
TCP_MC_ADDR_NOT_FOUND	-3021	multicast address not found
TCP_OUT_OF_PORTS	-3022	out of ports
TCP_NET_DOWN	-3023	network is down (send failed)
TCP_NET_UNREACH	-3024	network unreachable (keepalive failed)
TCP_OUT_OF_DCUS	-3025	out of DCUs (packets)
TCP_OPTPARAM_INVALID	-3026	option parameter is invalid
TCP SOCK NOT CONNECTED	-3027	socket is not connected
TCP_RTIP_NOT_INIT	-3028	RTIP not initialized (i.e. xn_rtip_init not called)
TCP_INVALID_SOCKET	-3029	invalid socket descriptor
TCP_NUM_DEVICE	-3030	not enough devices
TCP_OP_NOT_SUPPORTED	-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_SUPPORT	-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

If the TCP user interface is used in Lars, additional Windows Socket error codes can be returned. Here, it is important to note that these are negative and have an offset of 4000. Ex.: 14040 => WSA 10040 (WSAEMSGSIZE)

Details: <http://msdn.microsoft.com/en-us/library/windows/desktop/ms740668%28v=vs.85%29.aspx>

2.7.2 How to Use OS Functions

To use the OS TCP USER FUNCTIONS, a pointer named `lsl_tcp_user` of type `LSL_TCP_USER` must be defined and declared in “`lsl_st_tcp_user.h`”.

The pointer must be named exactly as stated above, as the macros declared in the `lsl_st_tcp_user.h` file use this pointer to call the OS functions.

Before the OS TCP USER FUNCTIONS can be used, the pointer must be initialized.

This is using the `OS_CILGet` OS macro declared in the global header file `rtos_interfaces.h`.

The name of the os tcp user interface is `TCP_USER`, this name is required for the first parameter of the `OS_CILGet` macro.

If the interface is available within the OS version used, the pointer will be initialized with the addresses of the OS functions. Otherwise a NIL-pointer will be returned by second parameter of the OS_CILGet macro. The pointer lsl_tcp_user of type LSL_TCP_USER can be defined as a global variable or a member variable.

To create a member variable of the type LSL_TCP_USER, the lsl_st_tcp_user.h header file must be included to the LASAL Class project and the global flag of the header file must be set to make the type LSL_TCP_USER available in the project.

Example

```
FUNCTION VIRTUAL GLOBAL MyClass::Init

  IF (_firstscan) THEN
    OS_CILGet("TCP_USER", #lsl_tcp_user);
  END_IF;

END_FUNCTION //VIRTUAL GLOBAL GetIP::Init
```

Once the OS TCP USER INTERFACE is available within the used operating system version, the OS_TCP_USER_VERSION macro can be used to check if a specified function is available.

See the requirements for each macro.

Example

```
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 FUNCTIONS

2.7.3.1 OS_TCP_USER_ACCEPT

Accept a connection from any remote host.

```
VAR_INPUT
  socket      : DINT;
  timeout_ms : UDINT;
END_VAR
VAR_OUTPUT
```

```
    retval      : DINT;
END_VAR;
```

Transfer parameters		Type	Description		
socket		DINT	Socket number		
timeout_ms	UDINT		0	Function does not block	
			>0	Timeout in ms, function blocks	
Return parameters		Type	Description		
retval		DINT	≥0	Socket number assigned to this connection	
				<0 Error code	

If timeout_ms is set >0, the function is blocked until the connection is established, an error has occurred or if the function takes to long and the timeout is expired.

If timeout_ms is set to 0, the function returns TCP_NOT_READY until the connection is established or an error has occurred (the function must be queried).

Example

```
retval      : DINT;
newssocket  : DINT;

retval := OS_TCP_USER_ACCEPT(socket, 0);
if retval < 0 & retval >> TCP_NOT_READY then
  ...error
else
  newssocket := retval; ...newssocket accepted ... ready to send or receive data
end_if;
```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.2 OS_TCP_USER_CLOSESOCKET

Close an existing socket.

```
VAR_INPUT
    socket      : DINT;
    type        : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Transfer parameters	Type	Description	
socket	DINT	Socket number	
type	UDINT	0	Hard close
		>0	Soft close
Return parameters	Type	Description	
retval	DINT	0	Function successful
		<0	Error code

A hard close closes the socket immediately and any unsent data will be lost.

A graceful close closes the socket but outstanding (queued) data will be sent.

Example

```
retval : DINT;

retval := OS_TCP_USER_CLOSESOCKET(socket, 0);
if retval < 0 then
    ...error
else
    ...o.k.
end_if;
```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.3 OS_TCP_USER_CONNECT

Establishes a connection to a specified socket

```

VAR_INPUT
    socket      : DINT;
    localport   : UDINT;
    IPAddress  : ^CHAR;
    port        : UDINT;
    timeout_ms : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;

```

Transfer parameters		Type	Description	
socket		DINT	Socket number	
localport		UDINT	Local port number (if 0, a unique port number is assigned)	
IPAddress		^CHAR	Destination IP address	
port		UDINT	Destination port number	
timeout_ms		UDINT	0 Function does not block >0 Timeout in ms, function blocks	
Return parameters		Type	Description	
retval		DINT	0 Function successful <0 Error code	

When the socket call is completed successfully, the socket is ready to send and receive data.

Each connection needs a different local port.

If timeout_ms is > 0, the function is blocked until the connection is established, an error has occurred or if the function takes to long and the timeout is expired.

If timeout_ms is set to 0, the function returns TCP_NOT_READY until the connection is established or an error has occurred (the function must be queried).

Example

```

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;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in Isl_st_tcp_user.h

2.7.3.4 OS_TCP_USER_GETCOMLINKPORT

Returns the port number that COMLink should use to wait for incoming connections.

```

VAR_OUTPUT
    port : UDINT;
END_VAR;

```

Return parameters	Type	Description
retval	DINT	≥0 Port number (1955 is the first Comlink port)

2.7.3.5 OS_TCP_USER_GETERRORSTRING

Get TCP user error message.

```

VAR_INPUT
    buffer      : ^CHAR;
    buflen      : UDINT;
    errvalue    : DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;

```

Transfer parameters	Type	Description
buffer	^CHAR	Pointer to a buffer that receives the error message
buflen	UDINT	Length of the buffer (≥ 90)
errvalue	DINT	Negative value returned by the function
Return parameters	Type	Description

retval	DINT	0 Function successful <0 Error code
--------	------	--------------------------------------------------------

Example

```

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;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.6 OS_TCP_USER_GETLINKSTATUS

The function provides information on the Ethernet interface iface in the form of bits.

```

VAR_INPUT
    iface      : DINT;
    pLinkStatus : ^UDINT;
END_VAR
VAR_OUTPUT
    retval : DINT;
END_VAR;

```

Transfer parameters	Type	Description
iface	DINT	Number of the Ethernet interface; 0 for the first interface, 1 for the second...
pLinkStatus	^UDINT	Pointer to the buffer for the link status

Return parameters	Type	Description	
retval	DINT	≥ 0	No error
		< 0	Error code

Meaning of the bits in the link status buffer:

Bit3 = 1	Transmission speed 1 Gigabit per second
Bit3 = 0 and Bit1 = 1:	Transmission speed 100 Megabits per second
Bit3 = 0 and Bit1 = 0:	Transmission speed 10 Megabits per second
Bit2 = 1	Full duplex mode
Bit2 = 0	Half duplex mode
Bit0 = 1	Connection established
Bit0 = 0	No connection established

In the header file `lsl_st_tcp_user.h`, several `#defines` are available for evaluating the bits.

<code>#define IP_LINK_STATUS</code>	0x01
	0 No connection
	1 Connection established
<code>#define IP_WIRE_SPEED</code>	0x02
	0 10 Mbits/s
	1 100 Mbits/s
<code>#define IP_DUPLEX_MODE</code>	0x04
	0 Half duplex
	1 Full duplex

2.7.3.7 OS_TCP_USER_GETPEERIP

Get IP address of the remote host that the socket is connected to.

```
VAR_INPUT
  socket      : DINT;
  sin_addr    : ^UDINT;
END_VAR
VAR_OUTPUT
```

```
    retval      : DINT;
END_VAR;
```

Transfer parameters		Type	Description
socket		DINT	Socket number
sin_addr		^UDINT	Pointer to an unsigned long variable that receives the IP address
Return parameters		Type	Description
retval		DINT	0 Function successful
			<0 Error code

Example

```
retval  : DINT;
ipaddr  : UDINT;

retval := OS_TCP_USER_GETPEERIP(socket, #ipaddr);
if retval < 0 then
    ...error
else
    ...o.k.
end_if;
```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.8 OS_TCP_USER_GETPEERPORT

Get port number of the remote host that the socket is connected to.

```
VAR_INPUT
    socket      : DINT;
    sin_port    : ^UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Transfer parameters	Type	Description
---------------------	------	-------------

socket	DINT	Socket number
sin_port	^UDINT	Pointer to an unsigned long variable that receives the port number
Return parameters	Type	Description
retval	DINT	0 Function successful <0 Error code

Example

```

retval  : DINT;
port    : UDINT;

retval := OS_TCP_USER_GETPEERIP(socket, #port);
if retval < 0 then
  ...error
else
  ...o.k.
end_if;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.9 OS_TCP_USER_GETSERVBYNAME

Get service info from service name and protocol name. This function is only available in the operating system RTK.

```

VAR_INPUT
  name      : ^CHAR;
  proto    : ^CHAR;
END_VAR
VAR_OUTPUT
  retval    : UDINT;
END_VAR

```

Transfer parameters	Type	Description
name	^CHAR	Pointer to a 0-terminated service name
proto	^CHAR	Optional pointer to a 0-terminated protocol name

Return parameters	Type	Description
retval	UDINT	<p>0 Function fails</p> <p>>0 Function succeeds, the port number of the service is returned</p>

If proto=NIL, OS_TCP_USER_GETSERVBYNAME returns the first service entry where name matches. Otherwise, getservbyname matches both the name and the proto.

The names are compared without regard to upper and lower case letters.

Service Name	Protocol Name	Description	Version
comlink_clnt	TCP	Base port number used for clients / e.g. alarms in lse-kernel	01.01.004 or later
online	TCP	Online serve	01.01.004 or later
comlink	TCP	Comlink command server	01.01.004 or later
comlink_refr	TCP	Comlink refresh server	01.01.004 or later
alarm	TCP	Alarms in lse-kernel	01.01.004 or later

Example

```
portnr      : UDINT;
portnr := OS_TCP_USER_GETSERVBYNAME("online", "tcp");
```

Requirements

Version: LasalOS 01.01.004 or later

LSL_TCP_USER STRUCT Member udVersion 3 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.10 OS_TCP_USER_GETSOCKIP

Get IP address of local host.

```
VAR_INPUT
  socket      : DINT;
  sin_addr    : ^UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;
```

Transfer parameters	Type	Description	
socket	DINT	Socket number	
sin_addr	^UDINT	Pointer to an unsigned long variable that receives the IP address	
Return parameters	Type	Description	
retval	DINT	0	Function fails
		>0	Function succeeds, the port number of the service is returned

Example

```

retval  : DINT;
ipaddr  : UDINT;

retval := OS_TCP_USER_GETPEERIP(socket, #ipaddr);
if retval < 0 then
  ...error
else
  ...o.k.
end_if;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.11 OS_TCP_USER_GETSOCKPORT

Get port number of local host.

```

VAR_INPUT
  socket      : DINT;
  sin_port    : ^UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;

```

Transfer parameters	Type	Description	
socket	DINT	Socket number	

sin_port	^UDINT	Pointer to an unsigned long variable that receives the port number
Return parameters	Type	Description
retval	DINT	0 Function successful <0 Error code

Example

```

retval  : DINT;
port    : UDINT;

retval := OS_TCP_USER_GETPEERIP(socket, #port);
if retval < 0 then
  ...error
else
  ...o.k.
end_if;
  
```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.12 OS_TCP_USER_IOCTLSOCKET

The tasks of this function are selected with the select_type parameter:

Parameter select_type is READ_AVAILABLE (1): the function provides the number of available bytes to read.

Parameter select_type is SET_SOCKET_MODE (3): the function sets the socket to the non-blocking (onoff=1) or blocking (onoff=0) mode.

The #defines READ_AVAILABLE and SET_SOCKET_MODE are available in the header file lsl_st_tcp_user.h.

```

VAR_INPUT
  Socket      : DINT;
  select_type : UDINT;
  onoff      : UDINT;
END_VAR
VAR_OUTPUT
  retval : DINT;
END_VAR;
  
```

Transfer parameters	Type	Description	
socket	DINT	Socket number	
select_type	UDINT	READ_AVAILABLE (1): Provides the number of bytes available to read For TCP, this is the number of bytes in the receive window. For UDP, it is the number of bytes in the first packet at the UDP input. WRITE_AVAILABLE (2): Only with RTK. Returns the number of bytes available for writing without blocking. For TCP, this is the size of the send window minus the number of bytes in the queue. For UDP, this is the maximum size of a transmission, i.e. the maximum fragment size if fragments are enabled, or the maximum amount of data that fits in a packet if fragmentation is disabled. SET_SOCKET_MODE (3): Sets the socket to non-blocking or blocking mode. See parameter onoff.	
onoff	UDINT	0	Blocking mode
		1	Non-blocking mode
Return parameters	Type	Description	
retval	DINT	≥0	No error
		<0	Error code

The #defines READ_AVAILABLE, WRITE_AVAILABLE and SET_SOCKET_MODE for the select_type parameter are available in the header file lsl_st_tcp_user.h.

2.7.3.13 OS_TCP_USER_IPINFO

Return information on the given network interface.

```

VAR_INPUT
  iface      : DINT;
  option     : DINT;
  pErg      : ^UDINT;
END_VAR
VAR_OUTPUT
  retval     : DINT;
END_VAR;
  
```

Transfer parameters	Type	Description	
iface	DINT	Network interface number to get information	
option	DINT	Information option	

pErg	^UDINT	Information
Return parameters	Type	Description
retval	DINT	0 Function successful <0 Error code

Option	pErg	OS Version
IP_OPT_ADDR	Binary IP address	≥ 01.01.004
IP_OPT_SUBNETMASK	Binary subnet mask	≥ 01.01.004
IP_OPT_ETHERNET_ADDR	The address of memory location where the Ethernet address is stored. To get the Ethernet address, cast an USINT pointer to this address.	≥ 01.01.021
IP_OPT_GATEWAY	Binary gateway address	≥ 01.01.127
IP_OPT_PORT	TCP port to which the TCP server listens	≥ 01.02.170
IP_OPT_DNS1	IP address of the first DNS server	≥ 01.02.190
IP_OPT_DNS2	IP address of the second DNS server	≥ 01.02.190
IP_OPT_DNS3	IP address of the third DNS server	≥ 01.02.190
IP_OPT_DNS4	IP address of the fourth DNS server	≥ 01.02.190

Example

```

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;

```

Requirements

Version: LasalOS 01.01.004 or later

LSL_TCP_USER STRUCT Member udVersion 4 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.14 OS_TCP_USER_LISTEN

Sets a socket to a state in which it waits for an incoming connection.

```

VAR_INPUT
  socket      : DINT;
  localport   : UDINT;
  backlogsize : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;

```

Transfer parameters	Type	Description
socket	DINT	Socket number
localport	UDINT	Local port number

backlogsize	UDINT	Maximum number of pending connections allowed
Return parameters	Type	Description
retval	DINT	0 Function successful <0 Error code

The local port number is the port number to which the clients can connect.

A socket in listen state cannot be used for send or receive data.

If backlog size contains an illegal value (less than 1 or greater than the number of max. connections), it is set to the number of maximum user connections allowed.

Example

```
retval : DINT;

retval := OS_TCP_USER_LISTEN(socket, 8000, 10);
if retval < 0 then
    ...error
else
    ...o.k. listen...
end_if;
```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.15 OS_TCP_USER_NREAD_AVAILABLE

Returns the number of bytes available to read.

```
VAR_INPUT
    socket      : DINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Transfer parameters	Type	Description
socket	DINT	Socket number
Return parameters	Type	Description

retval	DINT	>0	Number of bytes available to read
		0	No bytes available
		<0	Error code

Example

```

retval      : DINT;
BytesAvail  : DINT;

retval := OS_TCP_USER_NREAD_AVAILABLE(socket);
if retval < 0 then
    ...error
else
    BytesAvailable := retval;
    ...
end_if;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.16 OS_TCP_USER_PING

Sends a PING to the defined IP address.

```

VAR_INPUT
    ipaddr      : ^CHAR;
    bytes       : UDINT;
    ttl         : UDINT;
    wait        : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;

```

Transfer parameters	Type	Description
ipaddr	^CHAR	Pointer to the null-terminated ASCII String of the IP address
bytes	UDINT	Size of the PING package in bytes
ttl	UDINT	Life span (ttl - time to live) of the PING package in s

wait	UDINT	Timeout time (rt – round trip time) until response in ms
Return parameters	Type	Description
retval	DINT	<div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <p>≥0 Time in ms for the reply of the receiver</p> <p><0 Ping was not successful (error code)</p> </div> </div>

Recommended example values for the PING call:

The recommended values refer to the parameter values, used in the Command Line for the PING command.

bytes	32
ttl [s]	500
wait [ms]	2000

Example

```

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

Receive data at a socket.

```

VAR_INPUT
  socket      : DINT;
  buffer      : ^CHAR;
  buflen      : UDINT;
  flags       : UDINT;
  timeout_ms : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;

```

Transfer parameters	Type	Description
----------------------------	-------------	--------------------

socket	DINT	Socket number
buffer	^CHAR	Buffer for incoming data
buflen	UDINT	Length of buffer
flags	UDINT	Unused, must be set to 0
timeout_ms	UDINT	0 Function does not block >0 Timeout in ms, function blocks
Return parameters	Type	Description
retval	DINT	>0 Number of bytes available in the buffer 0 End of file (the connection has been closed by the remote) <0 Error code

If no data is available the function returns TCP_NOT_READY (if timeout_ms set to 0)

If timeout_ms is > 0 and no data is available, the function is blocked until data is available, an error occurs or the timeout expires.

Example

```

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;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.18 OS_TCP_USER_RECVFROM

Receive data at a socket (TCP and 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;
```

Transfer parameters		Type	Description
socket	DINT	Socket number	
buffer	^CHAR	Buffer for incoming data	
buflen	UDINT	Maximum number of bytes to receive	
flags	UDINT	Unused, must be set to 0	
timeout_ms	UDINT	0 Function does not block >0 Timeout in ms, function blocks	
pIPAddr	^UDINT	IP address of the sender	
pPort	^UDINT	Port number of the sender	
Return parameters		Type	Description
retval	DINT	>0 Number of bytes available in the buffer 0 End of file (the connection has been closed by the remote, TCP only) <0 Error code	

The function works like [OS_TCP_USER_RECV\(\)](#). The only difference is the information of the sender returned by pIPAddr and pPort.

Example Receiving an UDP Datagram

```
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;
```

Requirements

Version: LasalOS 01.01.104 or later

LSL_TCP_USER STRUCT Member udVersion 7 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.19 OS_TCP_USER_SELECT

This function tests the specified sockets for readiness to send or receive.

```

VAR_INPUT
  count      : DINT;
  Sockets    : ^DINT;
  select_type : UDINT;
  flags      : UDINT;
  timeout_ms : UDINT;
END_VAR
VAR_OUTPUT
  retval : DINT;
END_VAR;

```

Transfer parameters		Type	Description
count	DINT	Number of entries in the socket number array, to which psocket points	
Sockets	^DINT	Pointer to an array with the numbers of the sockets to test. The socket numbers are returned by the OS_TCP_USER_SOCKET() or OS_TCP_USER_SOCKET_EX() functions.	
select_type	UDINT	SELECT_OPT_READ: checks whether at least one socket is ready for reading. SELECT_OPT_WRITE: checks whether at least one socket is ready for writing. SELECT_OPT_EXCEPTION: checks whether a socket has an exception These values cannot be combined with one another.	
flags	UDINT	Not used, must be set to 0	
timeout_ms	UDINT	0 Function does not block >0 Timeout in ms, function blocks	
Return parameters		Type	Description
retval	DINT	>0 Total number of sockets ready <0 Error code TCP_NOT_READY: the number of ready sockets is null. TCP_INVALID_SOCKETNUM: no valid socket found. TCP_INVALID_PARAM: the pointer sockets is invalid or select_type is unknown.	

This function tests count sockets from sockets for readiness to send or receive. The calling task is blocked until an activity occurs in the specified socket or timeout_ms has elapsed.

The specified sockets are checked according to the select_type parameter, as to whether they ready for reading or writing, or whether an exception has occurred.

A server socket is considered ready for reading if a connection is pending that can be accepted with OS_TCP_USER_ACCEPT(). A client socket is ready for writing when a connection has been completed.

Exceptions do not mean errors. Errors are immediately reported when a defective system call is made. Instead, they include conditions such as an urgent message at a socket.

The timeout_ms parameter indicates the maximum waiting time in millisecond. Enter null as the time when you want to find out which sockets are ready without waiting when none are ready.

The normal return value of OS_TCP_USER_SELECT() is the total number of ready sockets. If the number of ready sockets is null, TCP_NOT_READY is returned. Which sockets are ready for the corresponding operation can be seen from the socket number array. If the content of an array element is unlike null, the socket is then ready. If the content of an array is 0, the socket is not ready.

If an error occurs, OS_TCP_USER_SELECT() returns a negative error code. The elements of the socket number array are null.

2.7.3.20 OS_TCP_USER_SEND

Send data to a socket.

```
VAR_INPUT
    socket      : DINT;
    buffer      : ^CHAR;
    buflen      : UDINT;
    flags       : UDINT;
    timeout_ms : UDINT;
END_VAR
VAR_OUTPUT
    retval     : DINT;
END_VAR;
```

Transfer parameters	Type	Description
socket	DINT	Socket number
buffer	^CHAR	Buffer to send
buflen	UDINT	Number of bytes to send

flags	UDINT	Unused, must be set to 0
timeout_ms	UDINT	0 Function does not block >0 Timeout in ms, function blocks
Return parameters	Type	Description
retval	DINT	>0 Number of bytes queued for sending 0 End of file <0 Error code

If the remote is not ready to receive data, the function returns TCP_NOT_READY if timeout_ms is set to 0 (must be queried).

If timeout_ms > 0, the function is blocked until the remote is ready to receive, an error occurs or the timeout expires.

The function is blocked until all data has been sent to the remote.

Large amounts of data should be sent in smaller blocks using several function calls.

Example

```

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;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.21 OS_TCP_USER_SENDTO

Send data to a socket.

```
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;
```

Transfer parameters	Type	Description	
socket	DINT	Socket number	
buffer	^CHAR	Buffer to send	
buflen	UDINT	Number of bytes to send	
flags	UDINT	Unused, must be set to 0	
timeout_ms	UDINT	0 Function does not block >0 Timeout in ms, function blocks	
ipaddr	UDINT	Destination IP address	
port	UDINT	Destination port number	
Return parameters	Type	Description	
retval	DINT	>0 Number of bytes queued for sending <0 Error code	

The function works like [OS_TCP_USER_SEND\(\)](#). The only difference is the destination IP address and port number which can be given by the parameters ipaddr and port. The parameters for the destination are only for UDP sockets and will be ignored for TCP sockets.

Example Sending an UDP Datagram

```
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;
```

Requirements

Version: LasalOS 01.01.104 or later

LSL_TCP_USER STRUCT Member udVersion 7 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.22 OS_TCP_USER_SETSOCKOPT

Set a socket option.

```

VAR_INPUT
  socket      : DINT;
  level       : DINT;
  option_name : DINT;
  option_value : ^CHAR;
  optionlen   : DINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;

```

Transfer parameters	Type	Description	
socket	DINT	Socket number	
level	DINT	Protocol level (SOL_SOCKET or IPPROTO_IP)	
option_name	DINT	Option to modify; see the table below for supported values	
option_value	^CHAR	Pointer to a buffer which contains values for the specified option	
optionlen	DINT	Length of data pointed to by parameter option_value	
Return parameters	Type	Description	
retval	DINT	0	Function successful
		<0	Error code

The following table is a summary of supported protocol levels and options.

level option_name	Default	option_value	Meaning
SOL_SOCKET SO_DELAYED_ACK	Enabled	Type DINT. 0 means turn off; non-0 means turn option on.	Delay sending TCP acknowledgment. In a stream of full-size packets, only every second packet will be acknowledged.
SOL_SOCKET SO_NAGLE	Enabled	Type DINT. 0 means turn off; non-0 means turn option on.	The Nagle Algorithm prohibits sending of small TCP packets (less than MSS) while there is any outstanding output data which has not been acknowledged.
SOL_SOCKET SO_BROADCAST	Enabled	Type DINT. 0 means turn off; non-0 means turn option on.	This option enables sending of broadcasts over a UDP socket and is supported starting from Salamander version 09.03.060.

SOL_SOCKET SO_REUSEADDR	Disabled	Type DINT. 0 means turn off; non-0 means turn option on.	<p>Enables the reuse of local addresses. When a server application binds a local address to a socket and then starts accepting connections on it (OS_TCP_USER_ACCEPT), the local address is bound to the local socket. If the server application is stopped and restarted, a subsequent attempt to bind the local address will fail. This is known as an "address in use" error. The reason for this is that the server socket is in the WAIT_STATE state. The socket remains in this state for two minutes and is then released so that the local address can be reused. The SO_REUSEADDR option can be used to force the possibility of reusing the local address before the two-minute time limit expires. To enable reuse, the option must be activated before calling OS_UDP_USER_BIND. Care should be taken when using this option as it makes TCP less reliable.</p>
IPPROTO_IP IP_MULTICAST_IF	Not set	Type UDINT. IP address in network byte order. The notes on the mc_group_addr parameter of the IP_ADD_MEMBERSHIP socket option apply with regard to the form of the transferred IP address.	Definition of the local Ethernet interface used to send multicast IP packets via the socket concerned. Please also note the information below on multicast IP addresses and the Internet Group Management Protocol (IGMP) or IGMP snooping.
IPPROTO_IP IP_MULTICAST_LOOP	Enabled	Type DINT. 0 means turn off; non-0 means turn option on.	<p>Deactivation or activation of the loopback of multicast packets sent via the socket to the local sockets. Normally, IP packets that are sent to a multicast group via a socket are also forwarded or returned to sockets in their own controller that are members of this multicast group, i.e. they wait for incoming IP packets sent to the address of this group. The option described allows you to deactivate this transfer and save resources as a result. However, this measure should be used with caution, as all traffic originating from a socket is forwarded to appropriately configured sockets on other controllers or computers, but not</p>

			to applications running on your own controller.
IPPROTO_IP IP_ADD_MEMBERSHIP	Disabled	Type IP_MC_REQ. Details see below.	This option is used to enable the reception of multicast IP packets for a specific multicast IP address. See below for details. Please also note the information below on multicast IP addresses and the Internet Group Management Protocol (IGMP) or IGMP snooping.

Option IP_ADD_MEMBERSHIP

This option is used to enable the reception of multicast IP packets for a specific multicast IP address or to join an IP multicast group as a recipient. The address of an IP_MC_REQ structure is transferred in option_value. The IP_MC_REQ data type is defined in the same header file as OS_TCP_USER_SETSOCKOPT (isl_st_tcp_user.h).

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 contains the address of the multicast group that the application wants to join. Its value must therefore be a valid multicast address and passed as an unsigned 32-bit integer. The arrangement of the bytes must follow the network byte arrangement (big endian) that applies independently of the hardware for TCP/IP. It is recommended to use the ConvertStrToBin() method of the OS interface class _IP to assign the correct value in the correct format to this parameter based on an IP address in the usual "dotted decimal notation". Alternatively, the value to be assigned can be calculated by multiplying the four integers of the IP address in the above notation by powers of the base 256 and adding them up. On little-endian platforms, e.g. Edge, Edge 2 and x86, the first integer is multiplied by 1, the second by 256, the third by 256² and the fourth by 256³ before the sum is calculated. On big-endian platforms, the above factors would have to be applied in reverse order.

If the developer has to define the multicast IP address used himself, the relevant standardization documents, in particular Section 6 of RFC 2365 (Administratively Scoped IP Multicast) of the IETF, must be observed. Some basic information on multicast IP addresses can also be found in the section "Using Multicast IP Addresses from the Administratively Scoped IPv4 Multicast Space" later in this document.

iface_addr is the address of the local Ethernet interface via which the system will receive the IP packets sent to the multicast group. The form of the transferred IP address is the same as for the parameter mc_group_addr. If the address is equal to IP_ADDR_ANY (0.0.0.0), the operating system attempts to select a corresponding Ethernet interface. However, it is recommended to always assign the IP address of the desired Ethernet interface to the member iface_addr. This avoids errors (e.g. -4019/No such device) when setting the IP_ADD_MEMBERSHIP option.

The OS_UDP_USER_BIND function must also be called for the multicast packets to be received correctly. It is sufficient to assign a combination of IP_ADDR_ANY and the desired port number to the socket.

Using Multicast IP Addresses from the Administratively Scoped IPv4 Multicast Space

The term "Administratively Scoped IPv4 Multicast Space" refers to a sub-area within the section of the IPv4 address space intended for multicast. RFC 2365 of the IETF entitled "Administratively Scoped IP Multicast" defines 239.0.0.0 as the lower limit and 239.255.255.255 as the upper limit of this sub-range. If no multicast IP address is specified, it is advisable to take into account the specifications of the RFC mentioned, in addition to any other standardization documents, and to select an IP address from the sub-range mentioned, specifically from the "IPv4 Local Scope". This section is part of the "Administratively Scoped IPv4 Multicast Space" and ranges from 239.255.0.0 to 239.255.255.255.

A multicast IP address from the "IPv4 Local Scope" transmitted as the destination address of an IP packet, together with the IP address of the sending interface transmitted as the source address, identifies a group of nodes in the same IP subnet as the multicast sender that are the exclusive recipients of the packet user data. Together with the combination(s) of IP addresses and subnet masks set on an IP interface, the source address contained in an incoming multicast IP packet determines its further processing. A socket configured to receive multicast packets with the destination address under consideration via the interface under consideration only receives a packet if its source IP address is in an IP subnet in which the IP interface under consideration also participates. Ethernet frames received at an Ethernet interface or multicast IP packets contained therein can therefore have the same multicast IP address as the destination address, but be intended for different IP subnets or groups of multicast receivers. This is the case if the IP addresses of the sender interfaces listed as source addresses in the IP packets are in different IP subnets.

When setting the IP_MULTICAST_IF socket option in a sender application, not only the local interface used to send multicast IP packets via the socket in question is determined,

but also the source IP address specified in the packets. In contrast, the interface address passed in `iface_addr` when setting `IP_ADD_MEMBERSHIP` by a receiver application only specifies the local interface used to receive multicast IP packets via the socket concerned. However, in order for the receiving socket to receive data from IP packets with a specific source IP address, the interface address passed in `iface_addr` does not necessarily have to be in the same IP subnet as the source IP address contained in the packets. Although the interface specified in this way must participate in the IP subnet defined by the sender using `IP_MULTICAST_IF`, this can also be provided by another IP address set on the interface. Multicast IP packets whose destination address matches the multicast IP address passed in `mc_group_addr` but whose source address is not in a corresponding IP subnet are destined for members of a group of multicast receivers defined by the same multicast IP address but in a different IP subnet, and are therefore not forwarded to the socket, even if they have been received via the interface specified in `iface_addr`.

If a sending interface participates in the same IP subnet as a receiving interface, but also represents a node of another IP subnet, i.e. has several combinations of IP address and subnet mask, there is therefore a certain risk that a multicast socket configured for receiving will not receive any data from a sending socket, even though the corresponding sockets could be configured for multicast without errors or the socket options mentioned could be set without errors. In such a case, it should be checked whether the interface IP address passed when setting the `IP_MULTICAST_IF` socket option in the send application is in the correct IP subnet, as it is also entered as the source address in the individual multicast IP packets.

IGMP Snooping

This term refers to a function of commercially available, mostly manageable Ethernet switches that serves to minimize the negative effects of multicast transmissions on the utilization and security of a network. The Internet Group Management Protocol (IGMP) is an IPv4-based protocol that allows routers and hosts within an IP subnet to exchange information about memberships in multicast groups. IGMP forms the basis for multicast in IPv4 networks by enabling the most targeted transmission of multicast data to those hosts within the network that need or have requested it. A so-called IGMP querier sends periodic requests to all possible multicast receivers within an IP subnet, among other things. This IGMP querier is usually a router that may have been selected from several routers available in the network. The IGMP membership queries may trigger IGMP membership reports, which inform hosts in the network about existing memberships in multicast groups. Hosts also send such IGMP membership reports unsolicited and without regular repetition if they want to become members of a multicast group.

With IGMP snooping, a switch processes the data of the IGMP packets sent by the hosts of a network and thus obtains information about the memberships of the hosts in various

multicast groups. This information enables the switch to forward multicast IP packets received on a port from a multicast sender only to those ports behind which there are hosts that wish to receive the packets in question. In this way, unnecessary network load caused by multicast IP packets can be avoided. However, IGMP snooping can lead to the transmission of multicast data to some of the recipients of a multicast group being interrupted for any length of time. Such interruptions can occur in an IP subnet in which there is no router or multicast querier and therefore the information from a switch about any multicast receivers on its ports is not regularly updated. In such a case, it is recommended to weigh up the possible effects on performance and security of the affected network against the possible effects of interference in the transmission of multicast data and to deactivate IGMP snooping if necessary. Excluded from this recommendation are multi-layer switches that themselves have the ability to perform IGMP membership queries in order to enable the use of IGMP snooping even without an IGMP-capable router in the Layer 2 network.

Example

```
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;
```

Requirements

Version: LasalOS 01.01.051 or later

LSL_TCP_USER STRUCT Member udVersion 5 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.23 OS_TCP_USER_SHUTDOWN

Disable receives and/or sends on a socket.

```
VAR_INPUT
    socket      : DINT;
    how        : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Transfer parameters	Type	Description	
socket	DINT	Socket number	
how	UDINT	0	Disable receives
		1	Disable sends
		2	Disable both
Return parameters	Type	Description	
retval	DINT	0	Function successful
		<0	Error code

The shutdown function does not close the socket. Any resources connected to the socket are not released until the closesocket function is called.

To ensure that all data is sent and received over a connected socket before it is closed, the shutdown function should first be used to close the connection before calling closesocket.

Example

```
retval : DINT;

retval := OS_TCP_USER_SHUTDOWN(socket, 2);
if retval < 0 then
    ...error
else
    ...shutdown...
end_if;
```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.24 OS_TCP_USER_SOCKET

Allocate a TCP socket.

```
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Return parameters	Type	Description
retval	DINT	<p>≥0 Socket number</p> <p><0 Error code</p>

If several Ethernet interface connections are available, data can be exchanged over all of them.

If [OS_TCP_USER_CONNECT\(\)](#) is used to connect to a remote host, the IP stack of the operating system decides over which Ethernet interface to send the packet.

When using [OS_TCP_USER_LISTEN\(\)](#) to listen for incoming connections from a remote host, the opened port is valid for all interfaces.

This feature is available in the operating system versions higher than 01.01.100.

If data should be sent out over a specified interface or incoming connections were only accepted for a specified interface, [OS_TCP_USER_SOCKET_EX\(\)](#) can be used.

Example

```
socket  : DINT;

socket := OS_TCP_USER_SOCKET();
if socket < 0 then
    ...error
else
    ...o.k.
end_if;
```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.25 OS_TCP_USER_SOCKET_EX

Allocate a TCP socket.

```
VAR_INPUT
  iface      : DINT;
END_VAR
VAR_OUTPUT
  retval     : DINT;
END_VAR;
```

Transfer parameters	Type	Description	
iface	DINT	Network interface to allocate the socket, 1 for the first interface, 2 for the second, ...	
Return parameters	Type	Description	
retval	DINT	≥0	Socket number
		<0	Error code

Example

```
socket : DINT;

socket := OS_TCP_USER_SOCKET_EX(2); // second interface
if socket < 0 then
  ...error
else
  ...o.k.
end_if;
```

Requirements

Version: LasalOS 5.52 or later

LSL_TCP_USER STRUCT Member udVersion 2 or later

Header: Declared in Isl_st_tcp_user.h

2.7.3.26 OS_TCP_USER_STRTOULONG

Convert a dotted-decimal format IP address to a unsigned long.

If a DNS server is configured in the autoexec.lsl file, the DNS name can also be entered as a CHAR string instead of the IP address (DNS lookup). It is then resolved and the IP address is converted into 32-bit binary format.

```
VAR_INPUT
```

```

buffer      : ^CHAR;
END_VAR
VAR_OUTPUT
  retval      : UDINT;
END_VAR;

```

Transfer parameters		Type	Description
buffer		^CHAR	Buffer that contains the IP address
Return parameters		Type	Description
retval		UDINT	IP address 0xFFFFFFFF if an error is detected

Example

```

ipaddr  : UDINT;
ipstr   : ARRAY[0..15] of CHAR;

_strncpy(#ipstr[0], "192.168.44.105");

ipaddr := OS_TCP_USER_STRTOULONG(#ipstr[0]);
if ipaddr = 0xFFFFFFFF then
  ...error
else
  ...o.k.
end_if;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.27 OS_TCP_USER_STRTOULONG_ASY

This function is used to convert a name into an unsigned long IP address. It works asynchronously and returns the result via the callback function pUserFct.

```

VAR_INPUT
  Buffer      : ^CHAR;
  pUserFct    : ^VOID;
  pUserParam  : ^VOID;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;

```

Transfer parameters		Type	Description	
Buffer	^CHAR	Address of a buffer with the pool address or IP address in decimal point notation (e.g.: us.pool.ntp.org)		
pUserFct	^VOID	Address of the callback function for returning the resolved IP address and a result code (0=Ok, -10 Error)		
pUserParam	^VOID	Pointer to any user data (e.g. the THIS pointer). This pointer is passed to the callback function		
Return parameters		Type	Description	
retval	DINT		0	No error
			<0	Error code
			-1	Pointer Buffer or pointer pUserFct or pointer pUserParam invalid

Example of a Correct Declaration of the Callback Function

```
FUNCTION GLOBAL __cdecl StrToUlongCallback
VAR_INPUT
    pUserParam : pVoid;      // pointer to any user data (e.g. the THIS pointer)
    IPAddress UDINT;        // resolved IP address
    retcode: DINT;          // result codes (0=Ok, -10 Error)
END_VAR
```

2.7.3.28 OS_TCP_USER_TOIP

Convert IP-Address from 4 INTs to a dotted decimal format IP address.

```
VAR_INPUT
    buffer      : ^CHAR;
    buflen      : UDINT;
    ID1        : UDINT;
    ID2        : UDINT;
    ID3        : UDINT;
    ID4        : UDINT;
END_VAR
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Transfer parameters		Type	Description	
buffer	^CHAR	Pointer to a buffer that receives the IP address		

buflen	UDINT	Length of the buffer
ID1	UDINT	First ID of IP address
ID2	UDINT	Second ID of IP address
ID3	UDINT	Third ID of IP address
ID4	UDINT	Fourth ID of IP address
Return parameters	Type	Description
retval	DINT	0 Function successful <0 Error code

Example

```

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;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.29 OS_TCP_USER_ULONGTOSTR

Convert an unsigned long IP address to a dotted decimal format.

```

VAR_INPUT
  buffer      : ^CHAR;
  buflen      : UDINT;
  IPAddr      : UDINT;
END_VAR
VAR_OUTPUT
  retval      : DINT;
END_VAR;

```

Transfer parameters	Type	Description	
buffer	^CHAR	Pointer to a buffer that receives the IP address	
buflen	UDINT	Length of the buffer	
IPAddr	UDINT	IP address in unsigned long format	
Return parameters	Type	Description	
retval	DINT	0	Function successful
		<0	Error code

Example

```

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;

```

Requirements

Version: LasalOS 5.28 or later

LSL_TCP_USER STRUCT Member udVersion 1 or later

Header: Declared in Isl_st_tcp_user.h

2.7.3.30 OS_UDP_USER_BIND

Bind a socket to a local port number and IP address.

```

VAR_INPUT
  socket    : DINT;
  ipaddr   : UDINT;
  port      : UDINT;
END_VAR
VAR_OUTPUT
  retval    : DINT;
END_VAR;

```

Transfer parameters	Type	Description

socket	DINT	Socket number
ipaddr	UDINT	Local IP address that should be bound to the given socket
port	UDINT	Local port number that should be bound (0 = random port number chosen by the IP stack of the operating system)
Return parameters	Type	Description
retval	DINT	0 Function successful <0 Error code

If the function is used with the IP address IP_ADDR_ANY and a specific port number, all packets sent to the port number are received.

If IP_ADDR_BROADCAST is used, only broadcast packets (IP address = 255.255.255.255) are received.

If the currently set IP address is used, only packets addressed to the specified address (IP and port number) are received.

OS_UDP_USER_BIND must be called in order to receive UDP datagrams. If OS_UDP_USER_BIND is used when sending UDP datagrams and IP_ADDR_ANY is specified for the IP address, the packet is sent on the corresponding Ethernet interface if more than one Ethernet interface is available. In this case, the IP stack of the operating system decides on which Ethernet interface the packet must be sent (depending on the configuration of the IP addresses). If IP_ADDR_BROADCAST is used, the packet is sent out on every Ethernet interface.

More information can be found in the examples of the functions [OS_TCP_USER_RECVFROM\(\)](#) and [OS_TCP_USER_SENDTO\(\)](#).



Starting from LasalOS 09.01.001:

With _ADD_ANY, as with IP_ADDR_BROADCAST, only broadcast packets are received. To receive normal UDP datagrams, the socket must be connected to a valid IP address.

Requirements

Version: LasalOS 01.01.104.or later

LSL_TCP_USER STRUCT Member udVersion 7 or later

Header: Declared in lsl_st_tcp_user.h

2.7.3.31 OS_UDP_USER_SOCKET

Allocate an UDP socket.

```
VAR_OUTPUT
    retval      : DINT;
END_VAR;
```

Return parameters	Type	Description	
retval	DINT	≥0	Socket number
		<0	Error code

When the UDP socket is no longer needed, [OS_TCP_USER_CLOSESOCKET\(\)](#) should be called to make it available again for other calls to OS_UDP_USER_SOCKET. A maximum of 6 UDP sockets can be opened simultaneously.

The [OS_UDP_USER_BIND\(\)](#) function can be used to bind the socket to a local port number and IP address. For more details see the description of the [OS_UDP_USER_BIND\(\)](#) function.

The [OS_TCP_USER_SENDTO\(\)](#) and [OS_TCP_USER_RECVFROM\(\)](#) functions can be used for sending and receiving UDP datagrams.

Example

```
socket : DINT;

socket := OS_UDP_USER_SOCKET();
if socket < 0 then
    ...error
else
    ...o.k.
end_if;
```

Requirements

Version: LasalOS 01.01.104.or later

LSL_TCP_USER STRUCT Member udVersion 7 or later

Header: Declared in Isl_st_tcp_user.h

2.7.4 Defines

Defines for OS_TCP_USER_IPINFO

2.7.4.1 #define IP_ADDR_ANY

Any address ([OS_UDP_USER_BIND\(\)](#))

2.7.4.2 #define IP_ADDR_BROADCAST

Broadcast address ([OS_UDP_USER_BIND\(\)](#))

2.7.4.3 #define IP_OPT_ADDR

To get the current IP Address.

2.7.4.4 #define IP_OPT_ETHERNET_ADDR

To get the Ethernet address.

2.7.4.5 #define IP_OPT_SUBNETMASK

To get the current subnet mask.

2.7.5 Enumeration of Ethernet Interfaces

Since a PLC can support more Ethernet interfaces it was agreed to number them following a fixed scheme.

- | | |
|-----|-------------------------------------------------------|
| 1 | First internal interface (ETH1) |
| 2 | Ethernet interface at the EWP (or ETH2 at the CCL912) |
| 3 | Varan interface |
| 4-6 | reserved |
| 7 | USB to Ethernet adapter |
| 8 | Second USB to Ethernet adapter |

2.7.6 Examples

Simple 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;
            elseif newsock >= 0 then
                tcp_step += 1;
            end_if;

    END_CASE;
END_FUNCTION
```

```

3:      retval := OS_TCP_USER_RECV(newsock, #data[0], sizeof(data), 0, 0);
      if retval <= 0 & retval <> TCP_NOT_READY then
          tcp_step := 5;
      elseif 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;
            elsif 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;
            elsif 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;
            elsif retval > 0 then
                tcp_step += 1;
            end_if;

    4:      OS_TCP_USER_CLOSESOCKET(sock, 0);
            sock := 0;
            tcp_step += 1;

    54:
END_CASE;

state:= READY;
END_FUNCTION //VIRTUAL GLOBAL TcpClient::CyWork
```

2.8 API XTimer

2.8.1 General

The „XTimer“ is a high-resolution timer that runs with a 1 μ s clock pulse. This timer is implemented in the Xilinx FPGA and is only available in the CIPC platform with Xilinx flash version F5 or higher.

2.8.2 Interface Functions XTIMER

Header files: Isl_st_xtimer.h

This section describes the timer interface. To use this interface, a pointer must first be obtained via OS_CILGET. The name of the interface is IXTIMER. If the same name is used for the pointer as in the following example, the predefined macros can be used to reduce and simplify coding.

2.8.2.1 XtimerInit

This function initializes the timer with an interval and callback function and can be called at any time to change the values. [XTimerStop\(\)](#) is executed automatically before changing the settings.

For a definition of the callback function see the following example. Register esi will be loaded with the value of parameter pThis on entry. Register eax will be loaded with a status code. If eax is 0 on entry, everything is OK. If an overflow occurs, it means at least one interrupt request was lost and bit 1 of eax is set. This can occur if the interrupt routine is very lengthy, the timer interval is very low or the machine is heavily loaded.

```
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)
```

Transfer parameters	Type	Description	
ulNum	UDINT	Counter index	
ulVal	UDINT	A counter value in μ s, the value ranges from 1 to 65535, which means an interval from 1 μ s to approximately 65 ms	
pThis	$^{\wedge}$ VOID	A THIS pointer of an object	
pCallback	$^{\wedge}$ VOID	A pointer to the desired callback function	
Return parameters	Type	Description	
retval	DINT	$\neq 0$	no error
		0	error

None of the parameters may be 0 (or NULL/NIL). After having executed this command, the timer is stopped.

The following should be considered to avoid timing problems:

- an interval of only few microseconds will lead to a high system load.
- a lengthy interrupt-routine will increase latency and prevent the system from running properly.
- an endless-loop or similar will hang the system.
- the command CLI and similar are not recommended at all.
- this function must be carefully designed.

Measuring minimum intervals lead to following results:

- CIPC 800 Mhz: 15 μ s
- CIPC 400 Mhz: 20 μ s

The DUT is a simple program, which executes the callback into LASAL Class that was online via ethernet and counted the interrupts.

Requirements

Version: Lasal OS V5.43 or later.

Example

```
#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

This function initializes the timer with a new interval value. This function can be called at any time to change the counter value. [XTimerStop\(\)](#) is executed automatically before changing the settings.

```

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)

```

Transfer parameters		Type	Description
ulNum		UDINT	Counter index
ulVal		UDINT	A counter value in μ s; the value ranges from 1 to 65536, which means an interval from 1 μ s to approximately 65 ms
Return parameters		Type	Description
retval		DINT	#0 no error

		0 error
--	--	---------

See [XTimerInit\(\)](#).

Requirements

Version: Lasal OS V5.43 or later.

Example

```
// change to 100µs
SXTIMER_XTIMERSETINTERVAL( 0, 100 );
```

2.8.2.3 XTimerSetMode

Determines the timer mode.

This function can be used to determine whether the timer is automatically restarted after each overrun or whether only one overrun should occur.

If the function is not called, the time is restarted after each overrun by default.

```
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)
```

Transfer parameters	Type	Description
ulNum	UDINT	Timer mode
usMode	USINT	SXTIMER_SINGLE_RUN: Single run; the timer runs over only one time after starting and is not restarted automatically SXTIMER_CONT_RUN: Continuous run; the time is restarted after each new restart

Return parameters	Type	Description
retval	DINT	≠0 no error 0 error

The timer mode can be set once or as required before timer is started.

Changing the time mode has no effect on a currently started timer.

By calling [XtimerInit\(\)](#), the time mode is reset to the default value SXTIMER_CONT_RUN.

Requirements

Version: starting from LasalOS Version 01.02.120, starting from interface version 01.01.001

Example

```
// set timer mode to single run
SXTIMER_XTIMERSETMODE( 0, SXTIMER_SINGLE_RUN );
```

2.8.2.4 XTimerStopAllUser

Stops all timers.

```
void XTimerStopAllUser(
  void
);
FUNCTION __CDECL GLOBAL P_XTimerStopAllUser
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR;

#define SXTIMER_XTIMERSTOPALLUSER()
  pSXTimerInfo^.XtimerStopAllUser
  $ P_XTimerStopAllUser()
```

Transfer parameters	Type	Description
none		
Return parameters	Type	Description
none		

Requirements

Version: LasalOS 5.43 or later.

Example

```
FUNCTION VIRTUAL GLOBAL DoNothing::StopAllTimer
  SXTIMER_XTIMERSTOPALLUSER(0);
END_FUNCTION //VIRTUAL GLOBAL DoNothing:: StopAllTimer
```

2.8.2.5 XTimerValue

Returns the current counter value.

```
void XTimerValue(
  unsigned long ulNum,
);
FUNCTION __CDECL GLOBAL P_XTimerValue
VAR_INPUT
  ulNum      : UDINT;
END_VAR
VAR_OUTPUT
  UIValue    : UDINT;
END_VAR;

#define SXTIMER_XTIMERVALUE(p1)
  pSXTimerInfo^.XtimerValue
  $ P_XTimerValue(p1)
```

Transfer parameters	Type	Description
ulNum	UDINT	Counter index
Return parameters	Type	Description
UIValue	UDINT	Returns the current counter value

Requirements

Version: LasalOS 5.43 or later.

Example

```
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 restarts at value 0 and continues counting.

```

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)

```

Transfer parameters	Type	Description
ulNum	UDINT	Counter index
Return parameters	Type	Description
none		

Requirements

Version: LasalOS 5.43 or later.

Example

```

FUNCTION VIRTUAL GLOBAL DoNothing::StopReset
    SXTIMER_XTIMERRESET(0);
END_FUNCTION //VIRTUAL GLOBAL DoNothing:: StopReset

```

2.8.2.7 XtimerStop

Stops the timer and disables its 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)

```

Transfer parameters	Type	Description
ulNum	UDINT	Counter index
Return parameters	Type	Description
none		

Requirements

Version: LasalOS 5.43 or later.

Example

```

FUNCTION VIRTUAL GLOBAL DoNothing::StopTimer
    SXTIMER_XTIMERSTOP(0);
END_FUNCTION //VIRTUAL GLOBAL DoNothing:: StopTimer

```

2.8.2.8 XTimerStart

Starts the timer and enables its 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)

```

Transfer parameters	Type	Description
ulNum	UDINT	Counter index
Return parameters	Type	Description
none		

none

Requirements

Version: LasalOS 5.43 or later.

Example

```
FUNCTION VIRTUAL GLOBAL DoNothing::StartTimer
  ICnt := 0;
  SXTIMER_XTMRSTART(0);
END_FUNCTION //VIRTUAL GLOBAL DoNothing:: StartTimer
```

2.9 Web Server

2.9.1 Instruction

The LasalOS Web Server is a simple Web Server application, which interprets user defined web pages written in HTML language. The LasalOS communicates with a remote Web browser using HTTP protocol over TCP.

2.9.2 Requirements

LasalOS	Version 01.01.50 or higher, websvr.dlm version 01.01.002 needed
Platform	All platforms with an Ethernet interface

2.9.3 Installation

The WEBSVR.DLM must be copied into the directory C:\LSLSYS

To use additional functions within LasalOS Web pages, WEBFNC.DLM must also be copied to the directory C:\LSLSYS (see chapter [Additional Functions](#), for more information)

The default root-directory for the Web Server is C:\

To change this directory, the environment variable WEBROOT has to be set before starting the Web Server.

The SETENV WEBROOT command can be used in the command line interface or the line can be added to the autoexec.lsl in order to start the Web Server automatically when the system starts.

The root-directory requires a trailing backslash

E.g.: SETENV WEBROOT C:\WEBROOT\

The directory C:\LSLSYS\HTML is reserved for LasalOS Web pages



The LasalOS Web Server can be started from the command line interface with WEBSVR START command or the autoexec.lsl can be edited to start the server automatically during start-up.

If the Web Server starts, the Web server DLL is loaded. If the Web Server is no longer needed, the DLL can be cleared with the WEBSVR STOP command.

The chapter on Command Line Interface (CLI) commands gives a more detailed description of each CLI command.

Two set up commands are available if security is required.

WEBSVR FILTER +address/mask

to add a host / network entry, which the Web Server compares on incoming connections.

WEBSVR AUTH ADD webpage realm username: password

to add an entry for user identification for the whole root-directory or several web pages.

The WEBSVR AUTH command cannot be used within the autoexec.lsl

See Chapter [Security](#) for more information on authentication!

2.9.4 Security

2.9.4.1 IP Address Authentication

Using the LasalOS CLI command [WEBSVR FILTER](#) or an API call from the application, a list of acceptable IP addresses can be set up. If no filter is set, the server accepts all IP addresses that can be reached through the network configuration. If a single filter is set, the Web server only accepts connections that are allowed through the filter setting. More than one filter can be set (max. 5, Web server DLL version 01.01.001).

Two different filter types can be entered:

HOST entries	The server only accepts connections from this IP-Address. The mask is therefore always 255.255.255.255.
--------------	------------------------------------------------------------------------------------------------------------

NETWORK entries	The server accepts all connections that exist within the network. With this setting, the mask varies from 255.255.255.255.
-----------------	-------------------------------------------------------------------------------------------------------------------------------

Example

Host entry: 10.100.100.100 / 255.255.255.255
10.100.100.101 / 255.255.255.255

The server only accepts a connection from remote host 10.100.100.100 and 10.100.100.101

Network entry: 10.100.100.0 / 255.255.255.0

The server accepts connections from remote hosts 10.100.100.X

If both host and network entries are set and the host entry is in the same network as the network entry, the host entry is ignored.

Example

Host entry: 10.100.100.100 / 255.255.255.255

Network entry: 10.100.100.0 / 255.255.255.0

In this example of filter configuration, the host entry is ignored.

There are two different ways to add an entry:

Operating System	These entries are added through the CLI WEBSVR FILTER command and are valid until the command deletes them. The entry is lost when the system is rebooted. The command can be added to the autoexec.lsl after the WEBSVR START command with each system start.
Application	These entries can be added by calling the API through the application. They are invalid and are cleared when the application is reset or by another API call.

2.9.4.2 Authentication with User Name and Password

With the LasalOS WEBSVR AUTH CLI command or an API call from the application, several web pages or entire web directories can be protected with user name and password authentication.

The LasalOS WEBSVR AUTH CLI command is not available in a DTC081 CPU containing an operating that supports the LasalOS Web Server.

The username and password are 64 bit encoded.

Three different types of authentication entries are available.

Default	These entries are generated by the OS at system startup and cannot be deleted. The user name and password can be changed however. A WEBPWD.TXT file is created that contains the entry information in an encoded format.
Operating System	These entries are generated when using the WEBSVR AUTH CLI command. The Operating System entries are also be saved in the WEBPWD.TXT file.
Application	These entries can be generated by calling the API from the application. The entry is valid after it's called but it is not stored in the WEBPWD.TXT file. After the application is reset, the entry is invalid and released.



The WEBPWD.TXT file is created and modified by the LasalOS only. If the content of the file is destroyed or changed externally, it can contain invalid entries that are loaded when the Web server is started. If the modified entry was a LasalOS default setting, all operating system entries and changes to any default value will be lost and the file will be created with the standard default entry with the next Web server startup.

2.9.5 Additional Functions

The LasalOS WebServer provides an additional DLL (WEBFNC.DLM) with several callback functions reserved for future use.

The directory C:\LSLSYS\HTML is reserved for Webpages using this function.

DO NOT USE THE DIRECTORY "C:\LSLSYS\HTML" FOR ANY HTML FILES!

2.9.6 Command Line Interface (CLI) Commands

2.9.6.1 WEBSVR, WEBSVR INFO

Gives information about available commands for the Web server, if the server is running and about the loaded DLLs and their versions.

Requirements

LasalOS: 01.01.050 or higher, not available for DTC081-IP

2.9.6.2 WEBSVR START

This command starts the Web server and can be executed manually or by the autoexec.lsl. It also creates the WEBPWD.TXT file if it does not exists or was modified or destroyed.

If the command fails, an error message will be printed to screen.

Possible reasons for an error

- Not enough memory
- Not enough free disk space to create the webpwd.txt file
- The WEBSVR.DLM (C:\LSLSYS) file is not available

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 or higher for DTC081-IP

2.9.6.3 WEBSVR AUTH

Optional and required arguments:

INFO

Shows information of all current authentication settings.

Example

```
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
```

The first entry is a LasalOS default entry, which is created by the operating system when the Web Server is started. This type of entry cannot be deleted, however, the username and/or password can be changed and an error 401 file added. The entry is saved in the WEBPWD.TXT file.

The second entry is an application entry, added by calling an API function. This type of entry is not saved in the WEBPWD.TXT file and the LasalOS remove the entry when the application is reset.

The third entry is an operating system entry and is added using the command WEBSVR AUTH ADD. The entry is saved in WEBPWD.TXT file and can be removed with WEBSVR AUTH REMOVE command if it is no longer required.

All entries in the WEBPWD.TXT file will be loaded during system startup.

If the WEBPWD.TXT file has been corrupted through modification or deletion, for example, all operating system entries are lost. A new WEBPWD.TXT file is created with the LasalOS default entries.

The default setting for this entry is 5, not including the LasalOS default entry. This setting can be increased using the variable WEBMAXAUTH.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, not available for DTC081-IP

**ADD <filename> <realm> <username:password>
<err401_file>**

Adds an authentication entry and stores it encoded in the WEBPWD.TXT file.

<filename>	Path of the web page or file name
<realm>	Protection partition, name of the protected resource or area on the server. The domain is shown in the dialog box of the browser, which requires the user name and password.
<username:password>	User name and password, both are required

Optional Parameter

<err401_file> User-defined web page sent to the browser on authentication failure

Example

```
WEBSVR AUTH ADD C:\WEBROOT Application Application:appl
```

All files in and below the directory C:\WEBROOT needs a username and password, but the username and password has only to be entered once as long as the browser remains opened.

```
WEBSVR AUTH ADD C:\WEBROOT\index.htm Application Application:index
```

In this case a username and password is only required for index.htm file. All other files in or below the directory C:\WEBROOT don't need a username and password.

```
WEBSVR AUTH ADD C:\WEBROOT\index.htm Application Application:index  
WEBSVR AUTH ADD C:\WEBROOT\ANYDIR Application Application:anydir
```

For index.htm a username and password is required, and all files in and below the directory

C:\WEBROOT\ANYDIR needs a username and a password.

```
WEBSVR AUTH ADD C:\WEBROOT\index.htm Application Application:index  
WEBSVR AUTH ADD C:\WEBROOT\test.htm Application Application:test
```

The file C:\WEBROOT\index.htm and C:\WEBROOT\test.htm needs a username and a password.

All other files in or below the directory do not need a username and password.

With this command it is also possible to change an entry.

To change an entry, the parameter filename and domain must be equal to the existing entry. The username and/or password can be changed or an error 401 file added. All entries can be changed, including the LasalOS default entry.

If an error 401 file is given in the entry, it must be available on the target or the web page, where the authentication required cannot be loaded.

Example

```
WEBSVR AUTH ADD C:\WEBROOT Application olduser:oldpass
```

Change above entry:

```
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
```

The entries are encoded and stored in the WEBPWD.TXT file.

The maximum number of entries can be set using the environment variable WEBMAXAUTH.

The default value is 5, not including the LasalOS default entries.

If the command fails, an error message will be printed to screen.

Possible Causes for an Error

- The list of entries is full, increase the value of the environment variable WEBMAXAUTH.
- The entry already exists.
- Not enough free disk space to store the entry in the WEBPWD.TXT file.
- Not enough memory.
- Invalid or insufficient parameters.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, not available for DTC081-IP

REMOVE <filename> <realm>

Removes an authentication entry and deletes it from the WEBPWD.TXT file.

<filename>	Path of the web page or file name added with the command ADD
<realm>	Protection partition, added with command ADD

Example

```
WEBSVR AUTH REMOVE C:\WEBROOT Application  
WEBSVR AUTH REMOVE C:\WEBROOT\index.htm Application
```

The username and password are not required to remove an entry. A LasalOS default entry cannot be deleted.

If the command fails, an error message will be printed to screen.

Possible Causes for an Error

- The entry does not exists
- Not enough free disk space
- Invalid or not enough parameter

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, not available for DTC081-IP

REMOVE ALL

Removes all authentication entries referenced by the operating system and application.

```
WEBSVR AUTH REMOVE ALL
```

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, not available for DTC081-IP

2.9.6.4 WEBSVR FILTER

Optional and required arguments:

INFO

Shows information of all current filters.

Example

```
WEBSVR FILTER INFO
```

Host/Net	Mask	Reference
10.100.100.0	255.255.255.0	Operating System
10.10.116.10	255.255.255.255	Application

The first entry is a network entry added by the operating system with the command WEBSVR FILTER +. This entry is static and can only be removed with the command WEBSVR FILTER -. A system reboot or shutdown also removes the entry.

The second entry is an application entry added by an API function call. This entry will be removed by an application RESET.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, not available for DTC081-IP

+<IPADDR>

Adds a host entry. The command is identical to the command below, but sets the mask automatically to 255.255.255.255.

+<IPADDR> <MASK>

Adds a host or network entry.

Example

```
WEBSVR FILTER +10.100.100.0 255.255.255.0          .. network entry
WEBSVR FILTER +10.10.116.10 255.255.255.255
WEBSVR FILTER +10.10.116.10                          .. host entry
```

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 or higher for DTC081-IP

-<IPADDR>

Removes a host entry.

Identical to the command below, but sets the mask to 255.255.255.255.

-<IPADDR> <MASK>

Removes a host or network entry.

Example

```
WEBSVR FILTER -10.100.100.0 255.255.255.0           .. network entry
WEBSVR FILTER -10.10.116.10 255.255.255.255
WEBSVR FILTER -10.10.116.10                         .. host entry
```

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 or higher for DTC081-IP

NONE

Removes all filter entries.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 or higher for DTC081-IP

2.9.6.5 WEBSVR STOP

This command stops the Web server. The Web server can only be stopped if there are no active connections.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 or higher for DTC081-IP

2.9.7 Environment Variables

2.9.7.1 WEBROOT

Sets the default root directory for the Web server. The value of the variable must have a trailing backslash.

To set an environment variable, the command [SETENV\(\)](#) has to be used.

Example

```
SETENV WEBROOT C:\WEBROOT\
```

The default root directory is C:\.

The WEBROOT variable must be set before the command WEBSVR START is executed. After the Web server is running, changes on the value of the variable has no effect.



The directory C:\LSLSYS\HTML is used by the web server for the LasalOS web pages.

2.9.7.2 WEBMAXAUTH

This variable is used to increase the number of operating system and application authentication entries. The default setting for this variable is 5, not including the LasalOS default authentication entries. The variable cannot be set to a value less than that of the default and must be set before the Web server is active.

Changes made while the Web server is running have no effect. To set an environment variable, the [SETENV\(\)](#) command is required.

Example

```
SETENV WEBMAXAUTH 10
```

2.9.7.3 WEBDEFPWD

This variable is used to replace a corrupted WebPwd.txt file. The user can decide whether to replace the authentication rights stored in the WebPwd.txt with default values when an error occurs during loading. The old Webpwd.txt is then renamed webpwd.cor and recreated. So that the corrupted WebPwd.txt is replaced, the environmental variable must be set to 1. In all other cases, the loading of the web server is aborted.

To set an environment variable, the command [SETENV\(\)](#) has to be used.

Example

```
SETENV WEBDEFPWD 1
```

2.9.8 Example Configuration

IPC, C-IPC with one Ethernet interface

IP address	10.10.116.10
Subnet mask	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 as root directory for the server.

Maximum number of operating system and application authentication entries 10.

Two network filter entries, allows connection from 10.10.x.x and 10.100.x.x.

Operating system authentication entries must be set up from the Command Line Interface (CLI).

```
WEBSVR AUTH ADD C:\WEBROOT\HTML TESTPAGE myname:mypass
```

2.9.9 LASAL OS Web Server API

To use the LasalOS Web server API, an `lsl_webserver` pointer of type `LSL_WEBSERVER_API` must be declared. The API `OS_CILGet()` function, declared in the `lsl_st_ssr.h` file, must be used to initialize the pointer.

If the pointer = NIL after calling the `OS_CILGet()` function, the interface is not available. This can happen if the OS version is too old (< 01.01.049), the Web server is not running or the Web server is not available on the Platform/CPU used. The name of the interface required for the first parameter of the `OS_CILGet()` function is `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;
```

If an API function is called when the `lsl_webserver` pointer is NIL, the application is stopped with an ACCESS EXCEPTION error.

2.9.9.1 OS_WEB_FindStringInBuffer

This function is used to locate a string in a buffer.

```
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)
```

Transfer parameters	Type	Description
pBuffer	^CHAR	Pointer to the buffer in which to search
pMatch	^CHAR	String for which to search
dBufLen	DINT	Length of the buffer
Return parameters	Type	Description
pRetval	^CHAR	When successful, a pointer to the beginning of the data is returned after the string pMatch. When an error occurs, a NIL pointer is returned.

The function searches the buffer for a string of the form “pMatch=...” with one of the following characters preceding: &, \0, \n

This will ensure that a search for values doesn't match ApplyValues.

The function is used to process post commands received from a browser with the API function [OS_WEB_GetLineFromBrowser\(\)](#).

Example

A search for “apply” in a buffer that contains “&apply=1” will return a pointer to 1.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 for DTC081-IP, Header-File: lsl_st_webserver.h

2.9.9.2 OS_WEB_GetErrorStringByValue

Returns a pointer to a LasalOS error message.

```
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)
```

Transfer parameters	Type	Description
dErrValue	DINT	Error value returned by an API function
Return parameters	Type	Description
pErrString	^CHAR	String for a LasalOS error message

Example

```
VAR
  pErrString     : ^CHAR;
  dErrValue      : DINT;
END_VAR

dErrValue := // ... API function call

if dErrValue <> 0 then
  pErrString := OS_WEB_GetErrorStringByValue(dErrValue);
end_if;
```

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 for DTC081-IP, Header-File: lsl_st_webserver.h

2.9.9.3 OS_WEB_GetLineFromBrowser

Is used to read the data sent by a browser from the user post callback function.

```
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)

```

Transfer parameters	Type	Description	
pHandle	pVoid	Handle passed to the callback function by parameter p_web_io_context	
ppBuffer	^pChar	Pointer to a pointer of type CHAR, to retrieve a pointer to the data	
dWait	DINT	The amount of time in seconds to wait for the data (Limit: 0-65 s)	
dType	DINT	Type of reading data	
		WEBS_G ET_BUF	Read the next available packet
		WEBS_G ET_LINE	Read to the next end of line
Return parameters	Type	Description	
dRetVal	DINT	≥ 0 Number of available bytes in the buffer <0 Negative error code	
		Use API function OS_WEB_GetErrorStringByValue to receive a pointer to an error message. See chapter Error Codes for more details.	

If dType WEBS_GET_LINE is used, the function reads data until “\r\n” is found or the buffer is full.

The buffer is 0-terminated and “\r\n” is saved. This function should be used within the user post callback to retrieve data sent by a browser.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 for DTC081-IP, Header-File: lsl_st_webserver.h

2.9.9.4 OS_WEB_InetStrToByte

Converts a string address to byte sequence.

```
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)
```

Transfer parameters	Type	Description
pStrAddr	^CHAR	Pointer to null-terminated string that contains the address
pByteAddr	^CHAR	Pointer to a byte array that retrieves the address in byte sequence
udSize	UDINT	Size of the byte array
Return parameters	Type	Description
pRetval	^CHAR	When successful, a pointer to the byte array is returned. If an error occurs, the return value is NIL. Possible causes for an error: Parameter pStrAddr = NIL, pByteAddr = NIL; udSize < 4;

The string, to which pStrAddr points, must contain the address in dotted format.

Example

```
VAR
  addr      : ARRAY [0..3] OF CHAR;
  pAddr     : ^CHAR;
END_VAR

pAddr := OS_WEB_InetStrToByte("10.100.100.11", #addr[0], sizeof(addr));
```

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 for DTC081-IP, Header-File: lsl_st_webserver.h

2.9.9.5 OS_WEB_RegisterGetCallback

With a special non-HTML code in the HTML source file, it is possible to allow the Web server to call an application callback function for a browser get command. OS_WEB_RegisterGetCallback installs a get callback function.

```
FUNCTION GLOBAL __CDECL P_RegisterGetCallback
VAR_INPUT
    pGetCallback : PVOID;
    pThis        : PVOID;
END_VAR;

#define OS_WEB_RegisterGetCallback(p1,p2)
    lsl_webserver^.pRegisterGetCallback
    $ P_RegisterGetCallback(p1,p2)
```

Transfer parameters	Type	Description
pGetCallback	PVOID	Callback function to be installed
pThis	PVOID	THIS pointer from the object of the class in which the callback is declared

Calling the function when one parameter = NIL will uninstall the callback.

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
```

Transfer parameters	Type	Description
pFncName	^CHAR	Pointer to a 0-terminated string that contains the name of the function referenced in the HTML source file
p_web_io_context	^LSL_WEB_IO_CONTEXT	Pointer to a LSL_WEB_IO_CONTEXT structure that contains a handle for exchanging data with the browser, a buffer for the data and the data length
pParam	^CHAR	Additional parameter, reserved
Return parameters	Type	Description

dRetVal	DINT	<p>Return value for the LasalOS</p> <table border="1"> <tr> <td>≤0</td><td>OS does not re-execute the function; this is useful when data must be sent only sending; when opening a web page, for example</td></tr> <tr> <td>≥1</td><td>OS will execute the function until dRetVal is set to ≤0; the function will be executed continuously with a delay of dRetVal milliseconds</td></tr> </table> <p>An application RESET stops the call of the callback functions.</p>	≤0	OS does not re-execute the function; this is useful when data must be sent only sending; when opening a web page, for example	≥1	OS will execute the function until dRetVal is set to ≤0; the function will be executed continuously with a delay of dRetVal milliseconds
≤0	OS does not re-execute the function; this is useful when data must be sent only sending; when opening a web page, for example					
≥1	OS will execute the function until dRetVal is set to ≤0; the function will be executed continuously with a delay of dRetVal milliseconds					



The callback function must be declared using `__CDECL`, otherwise it will not work.

If the Web server finds a reference of a function in the HTML source code, the callback function is executed.

Example

... HTML source code

```
<!--#exec cgi="/ApplTest.fn"-->
This line must be under comment!
```

The pFncName parameter of the GetCallback function points to the name ApplTest.fn.

The same callback function is executed for every function found in a web page.

The function names to which the pFncName parameter points must therefore be compared

in order to execute the correct code for each web page.



Prefix "OS_" is used by the webfnc DLL for LasalOS callbacks.

The function can send data directly to the browser; it is also possible to continuously send data to update the web page. The data must be placed in the LSL_WEB_IO_CONTEXT buffer.



The size of the buffer pointed to by `^LSL_WEB_IO_CONTEXT` is 1514 Bytes.

If more space is needed, a new pointer must be allocated. The original pointer must be saved and reset before the callback function is returned. The browser receives and shows everything sent by the function. The buffer can also contain Script Code, which the browser will execute. The send data to the browser the API function, [OS_WEB_SendLineToBrowser\(\)](#), must be used.



The callback function must not enter an infinite loop!

The chapter [Lasal OS Web server API - API Example](#) will show how the callback the functions are used.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 for DTC081-IP, Header-File: lsl_st_webserver.h

2.9.9.6 OS_WEB_RegisterPostCallback

The action attribute of the HTML object element form can be used to allow the Web server to call an application callback function for a browser post command. `OS_WEB_RegisterPostCallback` installs a post callback function.

```
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)
```

Transfer parameters	Type	Description
pPostCallback	PVOID	Callback function to be installed
pThis	PVOID	THIS pointer of the class object where the callback is declared

Calling the function when one parameter = NIL will uninstall the callback.

Postcallback

Type of the user post callback.

```
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
```

Transfer parameters		Type	Description
pFncName		^CHAR	Pointer to a null-terminated string that contains the name of the function referenced in the HTML source file
p_web_io_context		^LSL_WEB_IO_CONTEXT	Pointer to a LSL_WEB_IO_CONTEXT structure that contains a handle to exchange date with the browser, the data buffer and the data length
dLen		DINT	Number of bytes in the post command sent by the browser
Return parameters		Type	Description
dRetVal		DINT	Return value for the LasalOS. <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="flex: 1; background-color: #f0f8ff; padding: 5px; border-radius: 5px;"></div> <div style="font-size: 0.8em; padding: 0 5px;">≤0</div> <div style="flex: 1; background-color: #f0f8ff; padding: 5px; border-radius: 5px;"></div> <div style="font-size: 0.8em; padding: 0 5px;">OS does not execute the function again</div> </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="flex: 1; background-color: #f0f8ff; padding: 5px; border-radius: 5px;"></div> <div style="font-size: 0.8em; padding: 0 5px;">≥1</div> <div style="flex: 1; background-color: #f0f8ff; padding: 5px; border-radius: 5px;"></div> <div style="font-size: 0.8em; padding: 0 5px;">OS executes the function until dRetVal is set to ≤0</div> </div> <p>This function should not be called continuously.</p>



The callback function must be declared uainf __CDECL, otherwise it will not work.

If the Web server finds a reference of a function in the HTML source code, the callback function is executed. In comparison with the get callback function, where data is either sent once or continuously, the post callback is needed to receive data from the browser. Since that data is no longer available after being received, it is not practical to call this function continuously.

It is possible to update the page for certain amount of time within the post callback, but this should only be used to acknowledge data received from the browser. To read the data

sent by a browser, the API function [OS_WEB_GetLineFromBrowser\(\)](#) can be used. The API function [OS_WEB_SendLineToBrowser\(\)](#) can be used to send an acknowledgment.

Example

... HTML source code

```
<form action = "ApplPost.fn" method = "POST" name = "ApplTest">  
</form>
```

In this case the post callback, if installed, will be called with parameter pFncName = "ApplPost.fn".

For every function found in a web page, the same callback function is executed.

The function names to which the pFncName parameter points must therefore be compared

in order to execute the correct code for each web page.

Prefix OS_ is used by the webfnc DLL for LasalOS callbacks.



The data must be placed in buffer of LSL_WEB_IO_CONTEXT for it to be acknowledged

The size of the buffer to which ^LSL_WEB_IO_CONTEXT points is 1514 Bytes.



If more space is needed, a new pointer must be allocated.

The original pointer must be saved and reset before the callback function returns.

A typical application for the post commands is for user information (feedback, orders, ...).

The callback function must not loop forever!



The chapter [Lasal OS Web server API - API Example](#) will show how the callback the functions are used.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 for DTC081-IP, Header-File: lsl_st_webserver.h

2.9.9.7 OS_WEB_SendLineToBrowser

This function is used to send a line to a browser from the user get or post callback functions.

```
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)
```

Transfer parameters		Type	Description
pHandle	pVoid		Handle passed to the callback function by parameter p_web_io_context
dWait	DINT		The wait-time in seconds for previous sends to be acknowledged (Limit: 0-65 s)
dType	DINT		Type of the data sent WEBS_P Queue the data UT_QUE WEBS_P Send all queued data UT_SEN D
Return parameters		Type	Description
dRetval	DINT		0 Success <0 Negative error code

The function copies the bytes of data specified in p_web_io_context.^length_out from p_web_io_context.^buffer_out to the output queue. If no space is available, data already in

the queue is set to the browser. When the get or post callback functions is returned, all queued data is sent.

The size of the buffer is 1514 bytes.



If additional space is needed, the pointer to the buffer can be saved and a new pointer set. When the function is returned, the new pointer must be replaced by the old one.

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 for DTC081-IP, Header-File: lsl_st_webserver.h

2.9.9.8 OS_WEB_SetAuthentication

This function adds or changes an authentication entry from the application.

```
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)
```

Transfer parameters	Type	Description
pFilename	^CHAR	The path of the web page or file name
pRealm	^CHAR	Protection partition, name of the protected resource or area on the server. The domain is shown in the dialog box of the browser that requires the user name and password.
pUserPwd	^CHAR	User name and password are both required and must be separated by a colon

pErr401File	^CHAR	Optional, may be set to NIL if not needed
udFlags	UDINT	Flags for the type of operation. WEBS_A add an entry UTH_AD D WEBS_A remove an entry UTH_RE MOVE WEBS_A remove all entries UTH_RE MOVEAL L WEBS_A use in blocking mode UTH_BL OCKING
Return parameters	Type	Description
dRetval	DINT	0 Success #0 Error code

If the function is currently in use by another task, the WEBS_E_BUSY if WEBS_BL_BLOCKING function is not set. If set, the function blocks until it is released by the task that the function is currently using.

The flags ADD, REMOVE, REMOVEALL can be used or combined with the WEBS_AUTH_BLOCKING flag. If WEBS_AUTH_BLOCKING is set, the function blocks until it is released by the function currently using it. The duration of the function depends on the amount of entries in the authentication entry lists.

ADD, REMOVE and REMOVEALL cannot be combined.

The entries added with the API function are removed when the application is reset.

The entries are not saved in the WEBPWD.TXT file.

The entries will always be compared by pFilename and pRealm. To remove an entry, pUserPwd and pErr401File may be set to NIL. To change an entry, pFilename and pRealm must be equal to the existing entry that should be changed.

Example

```
// 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,
NIL,
WEBS_AUTH_REMOVE);

```

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 for DTC081-IP, Header-File: lsl_st_webserver.h

2.9.9.9 OS_WEB_SetBrowserList

This function sets a filter entry from the application.

```

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)

```

Transfer parameters	Type	Description
pHost	^CHAR	Pointer to a 0-3 array of type CHAR for the host/net
pNet	^CHAR	Pointer to a 0-3 array of type CHAR for the mask
udFlags	UDINT	Flags for the type of operation
Return parameters	Type	Description

dRetval	DIN	0 Success #0 Error code
---------	-----	----------------------------

If the function is currently in use by another task, the WEBS_E_BUSY if WEBS_BL_BLOCKING function is not set. If set, the function blocks until it is released by the task that the function is currently using.

The flags ADD, REMOVE, REMOVEALL can be used or combined with the WEBS_BL_BLOCKING flag. If WEBS_BL_BLOCKING is set, the function blocks until it is released by the function currently using it. The duration of the function depends on the amount of entries in the filter lists.

ADD, REMOVE and REMOVEALL cannot be combined.

The entries added with the API function are removed when an application is reset.

The API function [OS_WB_InetStrToByte\(\)](#) can be used to convert a string address to byte order.

This function does not check if entries already exist.

Example

```

VAR
  rc  : DINT;
addr  : ARRAY [0..3] OF CHAR;
  mask : ARRAY [0..3] OF CHAR;
END_VAR

OS_WB_InetStrToByte("10.100.100.100", #addr[0], sizeof(addr));
OS_WB_InetStrToByte("255.255.255.255", #mask[0], sizeof(mask));

rc := OS_WB_SetBrowserList(#addr[0], #mask[0], WEBS_BL_ADD);

```

Requirements

LasalOS: 01.01.050 or higher, Web server DLL version 01.01.002 or higher, 01.01.103 for DTC081-IP, Header-File: Isl_st_webserver.h

2.9.10 Quick Review

The amount of time the API functions needs is not defined and therefore unpredictable. It is not recommended to call this function from the cyclic tasks event or realtime tasks.

2.9.11 Error Codes

Define	Value	OS	Meaning
WEBS_E_NONE	0	≥ 01.01.049	no error
WEBS_E_TASKHANDLE	-1000	≥ 01.01.049	invalid task handle, error creating web server thread
WEBS_E_LISTFULL	-1001	≥ 01.01.049	list full (list of filter, authentication list)
WEBS_E_MEMORY	-1002	≥ 01.01.049	not enough memory
WEBS_E_NOT_FOUND	-1003	≥ 01.01.049	not found (filter entry, authentication entry)
WEBS_E_POINTER	-1004	≥ 01.01.049	invalid pointer (NIL)
WEBS_E_EXISTS	-1005	≥ 01.01.049	already exists (authentication entry)
WEBS_E_ACCESS	-1006	≥ 01.01.049	access denied (e.g.: try to remove a default auth. entry)
WEBS_E_BUSY	-1007	≥ 01.01.049	function currently in use (non-blocking mode)
WEBS_E_FLAGS	-1008	≥ 01.01.049	unknown flags (parameter udFlags)
WEBS_E_FNCDLL	-1009	≥ 01.01.049	error loading / unloading webfnc.dlm)

2.9.12 API Example

The following is a simple example that demonstrates the usage of API functions, as well as get and post callback functions.

Example of HTML and Structure Text source code.

Three HTML files

- index.htm
- AppICGI.htm
- Err401.htm

2.9.12.1 HTML Source Code: 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 Source Code: 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 SvrValuesTakeOver()"
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 SvrValuesTakeOver()
{
    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;
}
</script>
</body>
</html>
<!--#exec cgi="/ApplCGITest.fn"-->
```

2.9.12.3 HTML Source Code: 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 Project WebServer

```

Class:           WebServer

Servers:        ClassSvr      : DINT  (Read/Write)
                Svr1         : DINT  (Read/Write)
                Svr2         : DINT  (Read/Write)
                Svr3         : DINT  (Read/Write)

Methods:        Global         - Init
                Global         - Background
                Private        - WebServer  (Constructor)
                Private        - GetCallback  (__CDECL)
                Private        - PostCallback  (__CDECL)
                Private        - ConvertIToA
                Private        - ConvertAToI

Variables:      m_set_filter  : DINT;
                m_set_auth    : DINT;
                m_perrorstring :^CHAR;

Object Network: ONWebServer

Objects:        objWebServer - Background Time = 10

```

2.9.12.5 LASAL CLASS Project 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;
// 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_WB_SendLineToBrowser returns success
// dRetval = 500, OS waits 500 ms to the next call
dRetval := 500;
// send data to the browser
rc := OS_WB_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
    pFncName : ^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 _strcmp(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^ <> '&'amp;
            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';</scr
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::ConvertItoA //
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::ConvertItoA
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 Autoexec.lsl settings

```
...
SETENV WEBROOT C:\WEB\HTML\
WEBSVR START
...
```

The HTML source files index.htm, AppICGI.htm and Err401.htm have to be copied to C:\WEB\HTML\

2.9.13 Error, Warning and Info logging

With the Command Line Interface (CLI) instruction, SET DBGLEVEL WEBSVR, some info logging can be deactivated.

```
DBGLEVEL WEBSVR
```

- | | |
|---|-----------------------------------|
| 0 | no logging |
| 1 | error messages only (default) |
| 2 | error messages and debug messages |
| 3 | extended information |

Error logging can be activated with the autoexec.lsl at startup or manually through the Command Line Interface (CLI) command.

Example

```
SET DBGLEVEL WEBSVR 2
for error logging
```

2.9.14 Appendix

2.9.14.1 Types

```
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 Error Values

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

Filter Entry Flags

WEBS_BL_ADD	0x00000001	Add filter entry
WEBS_BL_REMOVE	0x00000002	Remove filter entry
WEBS_BL_BLOCKING	0x00000004	Blocking mode
WEBS_BL_REMOVE_ALL	0x00000008	Remove all entries

Authentication Entry Flags

WEBS_AUTH_ADD	0x00000001	Add authentication entry
WEBS_AUTH_REMOVE	0x00000002	Remove authentication entry
WEBS_AUTH_BLOCKING	0x00000004	Blocking mode
WEBS_AUTH_REMOVE_ALL	0x00000008	Remove all entries

Flags for Sending Data to a Browser

WEBS_PUT_QUE	1	Queue data
WEBS_PUT_SEND	2	Send all queued data

Flags for Reading Data from a Browser

WEBS_GET_LINE	1	Get line up to next new line
WEBS_GET_BUF	2	Get buffer

3 OS Interface Library Classes

3.1 Memory Library Classes

3.1.1 RamFile

The Ram file class is used to store data in a file. The data is stored in the file Fdata. If the file created is less than 64 kB, the contents are loaded in the RAM so that the read access is significantly faster. The write access is executed the same as normal data accessing. If the file is larger than 64 kB, data is read from or written to the file only.

3.1.1.1 Interfaces

Servers

m_udLength	Shows the length of the file data block (without data header).
FileNameHex	This number serves simultaneously as the data name and is different for each object.

Clients

Setup	Bit 0 Must be set to 1 to start the RamFile
	Bit 1 Must be set to 0 in order to edit a file
	Bit 2 1: activates the Checksum calculation
	Bit 3 1: the file is encoded
	Bit 4-15 Is reserved as an initialization value
Alarm	Set to 1 if the checksum doesn't correspond with the stored file at start up!
TaskObjectControl	Automatically linked to the operating system channel _TaskObjectControl

3.1.1.2 Global Methods

3.1.1.2.1 GetDataAt

This method is used in order to call data at any position. A pointer to the memory area is given from which the data should be stored after it has been read as well as the length and the position of the first byte in the desired data block.

The status of the data access is called using the method [GetFileState\(\)](#). When this method returns the 0 value, the GetDataAt() method is complete. For longer data blocks, this method can take a very long time. Therefore, it is recommended that the [GetDataAtBackground\(\)](#) method be used.

Transfer parameters	Type	Description
p_us_data		Pointer to data that has been read
du_size		Length of the date to be read
du_at		Position from which the data should be read
Return parameters	Type	Description
ret_code		Returns the actual status

3.1.1.2.2 GetDataAtBackground

This method works exactly the same as the [GetDataAt\(\)](#) method except that a background task is assigned and the entire file access is processed in the background. The status of the data access is called using the [GetFileState\(\)](#) method. When this method returns the 0 value, the GetDataAtBackground() method is complete.

Transfer parameters	Type	Description
p_us_data		Address to where the data is written
ud_size		Length of the data to be read
ud_at		Position from where the data should be read
Return parameters	Type	Description
ret_code		Returns the actual status

3.1.1.2.3 [GetFileState](#)

Access to large files can often take long. In order to avoid accessing the same file simultaneously and possibly reading false data, the next data access is only possible after the last is complete.

To determine whether the last file access is complete, the actual status can be queried with help from this function. When this method returns the value 0, the file can be accessed with the next method.

Return parameters	Type	Description	
State		0	no file access
		1	file in process

3.1.1.2.4 [GetUserVersion](#)

With help from this method, the user can read their own user version from the header file.

Return parameters	Type	Description	
Version	WORD	User version	

3.1.1.2.5 [SetDataAtBackground](#)

This method works exactly the same as the [SetDataAt\(\)](#) method except that a background task is assigned and the complete data access is processed in the background.

The status of the data access is called using the [GetFileState\(\)](#) method. When this method returns the value 0, the [SetDataAtBackground\(\)](#) method is complete.

Transfer parameters	Type	Description	
p_us_data		Pointer to the data that should be written	
ud_size		Length of the data to be written	
ud_at		Position at which the data should be written	
Return parameters	Type	Description	
ret_code		Returns the actual status	

3.1.1.2.6 SetDataAt

By calling this method, the transferred data is written to the file. A pointer to the data, which should be written to the file, must be given as well as the length and position of the first byte in the data block to be written.

The status of the data access is called using the [GetFileState\(\)](#) method. When this method returns the 0 value, the SetDataAt() method is complete.

With longer data blocks this function can take a very long time. Therefore it is recommended that the [SetDataAtBackground\(\)](#) method be used.

Transfer parameters		Type	Description
p_us_data			Pointer to the data that should be written
ud_size			Length of the data to be written
ud_at			Position at which the data should be written
Return parameters		Type	Description
ret_code			Returns the actual status

3.1.1.2.7 SetSize

This method is used to change the file size and should be called only once with the maximum file size. Cyclically calling this method reduces the life span of the CF significantly, as the size of the FAT and directory must be updated with every change.

When an available file with a valid name is expanded, the added data area is written with a 0! If the data is reduced, the file is simply cut and the data at the end of the file is lost.

The status of the file access is called using the [GetFileState\(\)](#) method. When a value of 0 is returned, the Setsize() method is complete.

For files larger than 50 kbytes, this function can take a very long time; the [SetSizeBackground\(\)](#) function should therefore be used.

Transfer parameters		Type	Description
ud_size			The new length of the file data block
Return parameters		Type	Description
ret_code			Returns the actual status

3.1.1.2.8 SetSizeBackground

The method works exactly the same as [SetSize\(\)](#) except a background task is assigned and the entire file is processed in the background. The status of the file access is called using the [GetFileState\(\)](#) method. When this method returns a value of 0, the SetSizeBackground() method is completed.

Transfer parameters	Type	Description
ud_size		The new length of the file data block
Return parameters	Type	Description
ret_code		Returns the actual status

3.1.1.2.9 SetUserVersion

With help from this method, the user can write their own user version in the header file.

Transfer parameters	Type	Description
Version	WORD	User version

3.1.1.3 Checksum Calculation

In order to detect corruption in a file, a checksum calculation can be created in the setup server. For this, bit 2 must be set (as initialization value). If the Checksum Calculation is to be deactivated, bit 2 must be reset (as initialization value).

This function increments the file access and shortens the life span of the CF.

3.1.1.4 Encoding

So that the file is not accessible to every user, it can be encoded using the setup server. For this, bit 3 must be set (as initialization value). If the file is to be un-encoded, bit 3 must be reset (as initialization value).

3.1.2 RamFileRingBuffer

The RamFileRingBuffer() class is used to store data records in a file. The data is stored in the file Fdata. If the file created is less than 64k, the contents are loaded in the RAM so that the read access is significantly faster. The write access is executed the same as normal data accessing. If the file is larger than 64 kB, data is read from or written to the

file only. The number and length of the data record is defined at creation and the updated data stored at the end. If the file end is reached, the oldest value is replaced with the current ensuring that the most current data record is stored in the file.

RamFileRingBuffer	
RamFileRingbuffer0	
Setup	m_udLength
2#00000000000000000000000000000000...	60048
Alarm	FileNameHex
0	16#CA47918F
	Entries
	600
	EntryLength
	100

3.1.2.1 Interfaces

Servers

m_udLength	Shows the length of the file data block (without data header).
FileNameHex	This number serves simultaneously as the data name and is different for each object.
Entries	Shows the maximum number of entries in a file.
EntryLength	Shows the length of an entry.

Clients

Recordup	Bit 0	Must be record to 1 to start the RamFile
	Bit 1	Must be record to 0 in order to edit a file
	Bit 2	1: activates the Checksum calculation
	Bit 3	1: the file is encoded
	Bit 4-15	Is reserved as an initialization value
Alarm	Record to 1 if the checksum doesn't correspond with the stored file at start up!	

3.1.2.2 Global Methods

3.1.2.2.1 GetDataRB

This method is used to call a desired data record. A pointer to the memory area is given in which the accessed data should be stored as well as the position of the data record. The actual data record is located at position 0; the next older data is at position 1 and so on.

The status of the file access is called using the [GetFileState\(\)](#) method. When this method returns the value 0, the GetDataRB() method is complete.

With longer data blocks, this function can require a lot of time. Therefore it is recommended that the [GetDataRBBackground\(\)](#) be used.

Transfer parameters	Type	Description
p_us_data		Pointer to the data record that has been read
ud_position		Position of entry (0 = current entry)
Return parameters	Type	Description
ret_code		Returns the actual status

3.1.2.2.2 GetDataRBBackground

This method works exactly the same as the [GetDataRB\(\)](#) method except that a background task is assigned and the entire file access is processed in the background. The status of the file access is called with the [GetFileState\(\)](#) method. When this method returns the value 0, the GetDataRBBackground() is complete.

Transfer parameters	Type	Description
p_us_data		Pointer to the accessed data
ud_position		Position of entry (0 = current entry)
Return parameters	Type	Description
ret_code		Returns the actual status

3.1.2.2.3 GetLastDataRB

This method is used to call several data records. A pointer to the memory area in which the accessed data should be stored as well as the number of accessed data records. The last data record is always read.

The status of the file access is called using the [GetFileState\(\)](#) method. When this method returns the value 0, the [GetDataRB\(\)](#) method is complete. With longer data blocks, this function can require a lot of time. Therefore, it is recommended that the [GetDataRBBackground\(\)](#) be used.

Transfer parameters		Type	Description
p_us_data			Pointer to the accessed data
ud_anzahl			Number of entries to be read (10 = last entry)
Return parameters		Type	Description
ret_code			Returns the actual status

3.1.2.2.4 GetLastDataRBBackground

This method works exactly the same as the [GetLastDataRB\(\)](#) except that a background task is assigned and the entire file access is processed in the background. The file access status is called using the [GetFileState\(\)](#) method. When this method returns the value 0, the [GetDataRBBackground\(\)](#) method is complete.

Transfer parameters		Type	Description
p_us_data			Pointer to the accessed data
ud_anzahl			Number of entries to be read (10 = last entry)
Return parameters		Type	Description
ret_code			Returns the actual status

3.1.2.2.5 GetNumberOfValidEntries

This method gives the number of the current number of (valid) data records. If the ring buffer is written to for the first time, this method returns the number of possible entries (all data records are now valid).

Return parameters		Type	Description
ud_number			Number of valid data records

3.1.2.2.6 RecordDataRB

By calling this method, a new data record is written to the file. Only the pointer to the data record, which should be written to the file, is required as a parameter. The file access is called using the [GetFileState\(\)](#) method. When this method return the value 0, the RecordDataRB() method is complete. With longer data blocks, this method can require a lot of time. Therefore it is recommended that the [RecordDataRBBackground\(\)](#) method be used.

Transfer parameters	Type	Description
p_us_data		Pointer to the data record to be written
Return parameters	Type	Description
ret_code		Returns the actual status

3.1.2.2.7 RecordDataRBBackground

This method works exactly the same as the [RecordDataRB\(\)](#) method except that a background task is assigned and the entire file access is processed in the background. The file access is called using the [GetFileState\(\)](#) method. When this method returns the value 0, the RecordDataRBBackground() method is complete.

Transfer parameters	Type	Description
p_us_data		Pointer to the data record that should be written
Return parameters	Type	Description
ret_code		Returns the actual status

3.1.2.2.8 RecordRingbufferSize

This method is used to change the file size and should be called only once with the maximum file size. If this method were called cyclically, the life span of the CF would be drastically reduced, as the size of the FAT and Directory must be updated with every change. If the file has does not already exist, it is created by calling this method with the transferred values.

If the file is already available and the new length of an entry (ud_length) is not the same as in the existing file, it is regenerated and all of the old data is lost! If the length of an entry (ud_length) remains unchanged, the data is simply resorted. If the file is smaller, the number of entries is reduced and the oldest entries are lost. When the number of entries is enlarged all data is retained; the new entries are record to 0.

The status of the file access is called using the [GetFileState\(\)](#) method. When this method returns the value 0, the RecordRingbufferSize() method is completed. For larger files (larger than 50 kbytes), this function can take a very long time. Therefore the method [RecordRingbufferSizeBackground\(\)](#) should be used.

Transfer parameters	Type	Description
ud_entries		The number of maximum entries in the file
ud_length		The length of an entry
Return parameters	Type	Description
ret_code		Returns the actual status

3.1.2.2.9 RecordRingbufferSizeBackground

The method works exactly the same as [RecordRingbufferSize\(\)](#) except a background task is assigned and the entire file is processed in the background. The status of the file access is called using the [GetFileState\(\)](#) method. When this function returns a value of 0, the RecordRingbufferSizeBackground() method is completed.

Transfer parameters	Type	Description
du_entries		The number of maximum entries in the file
ud_length		The length of an entry
Return parameters	Type	Description
ret_code		Returns the actual status

3.1.2.2.10 RecordUserVersion

With help from this method, users can write their own user version to the header file.

Transfer parameters	Type	Description
Version	WORD	User version

3.1.2.3 Checksum Calculation

In order to detect corruption in a file, a checksum calculation can be created in the setup server. For this, bit 2 must be set (as initialization value). If the Checksum Calculation is to be deactivated, bit 2 must be reset (as initialization value). This function increments the file access and shortens the life span of the CF.

3.1.2.4 Encoding

So that the file is not accessible to every user, it can be encoded using the setup server. For this, bit 3 must be set (as initialization value). If the file is to be un-encoded, bit 3 must be reset (as initialization value).

4 LARS

4.1 LARS

4.1.1 Introduction

LARS stands for Lasal Runtime System and is a tool of the LASAL operating system LasalOS and runs on the Microsoft Windows system. LARS can run LASAL applications with limited real-time and hardware requirements.

4.1.2 System Requirements

LARS can run on an IPC or a standard PC and requires the following hard and software:

- Windows 2000, Microsoft Windows XP or newer
- Pentium 300 MHz microprocessor or higher
- A minimum of 64 MB RAM
- SVGA 800 x 600 screen resolution or higher

The hardware requirements (processor RAM and memory space) also depend on the memory configuration of LARS, the number of running instances on a system, the amount of memory space to store LASAL projects and the required resources, which the running application uses in LARS.

4.1.3 Installation

Run LarsXX_S.exe and follow the instructions that appear on the screen. X stands for the serial number.

4.1.4 Configuration

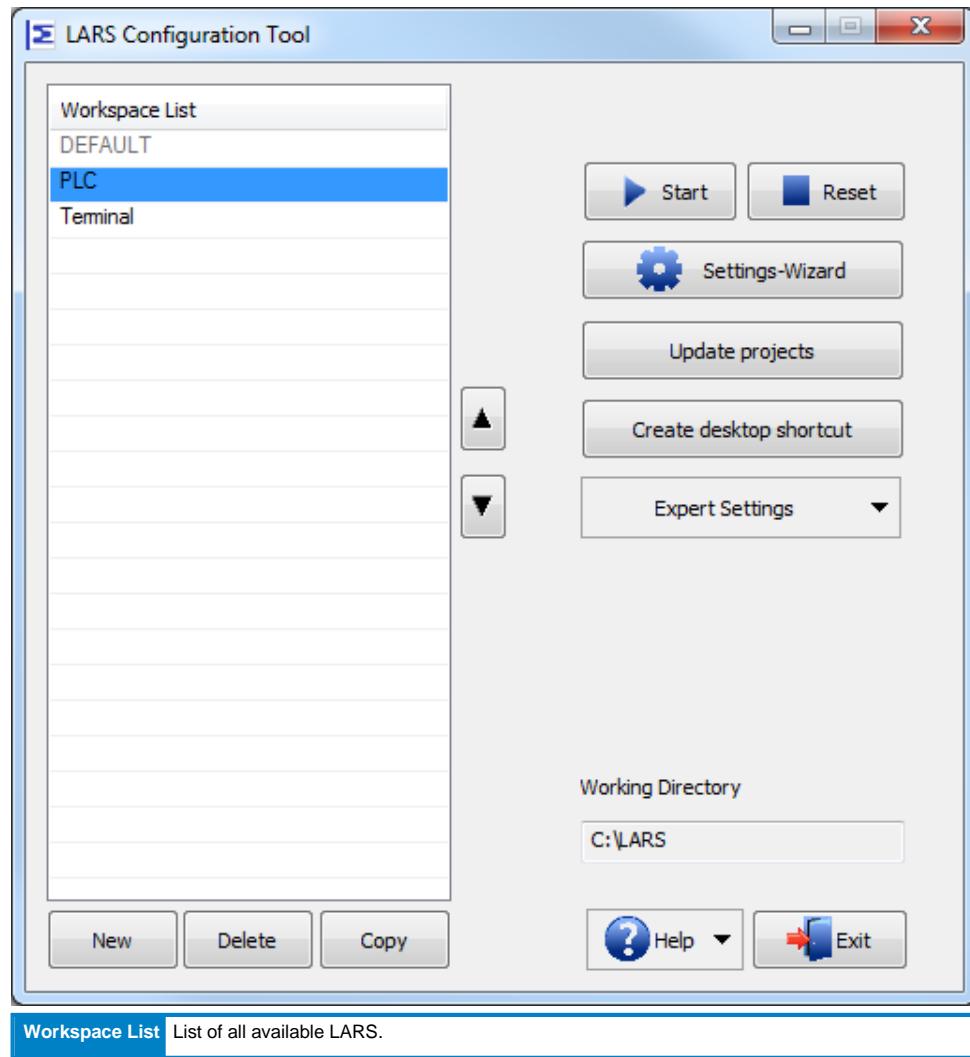
LARS must be started with a workspace. In no workspace is specified, a search is made for a default workspace.

Command Line Parameter

```
/c"<XML-config-file path>"  
/n<Name of the LARS workspace >  
/s<initial-screenmode> (screen mode = WIN or FULL, default = WIN)
```

4.2 LARSConfigTool

With the LarsConfigTool, workspaces for LARS can be created and parameterized. The information is stored in %APPDATA%\lasalos2.xml.

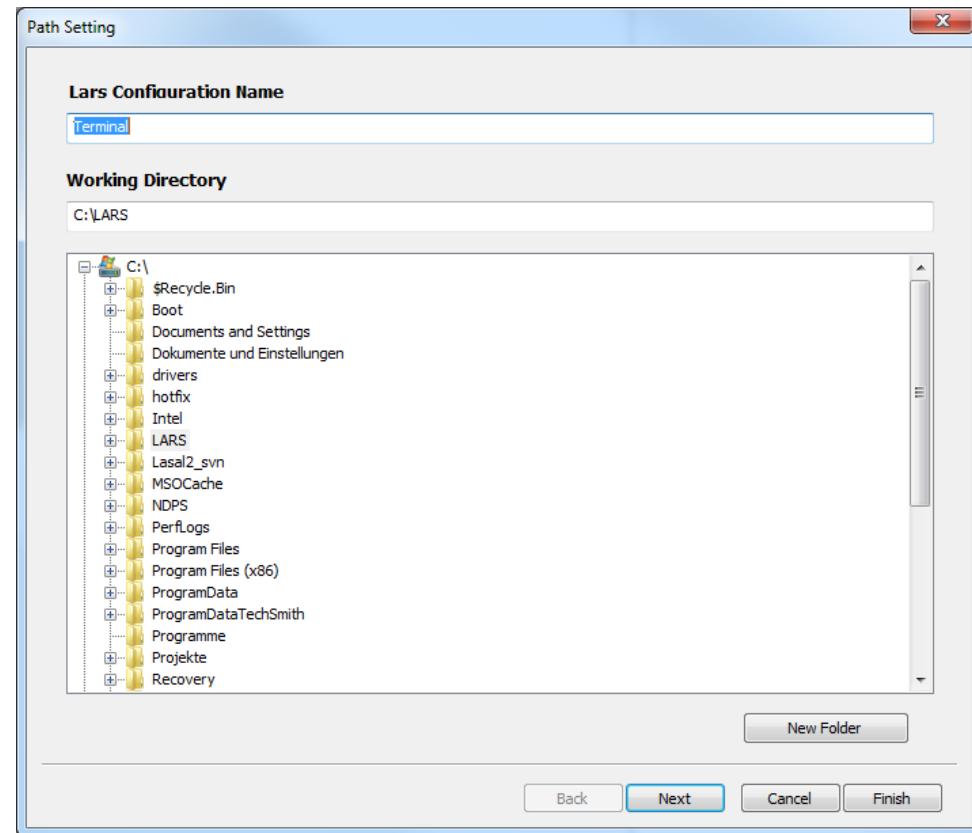


New	Creates a new workspace.
Delete	Deletes the selected workspace.
Copy	Copies the selected workspace.
Start	Starts the selected LARS project.
Reset	Stops the opened LARS project.
Settings Wizard	Starts the configuration of the settings in a wizard. See Settings Wizard
Update projects	LARS is started with the selected workspace, the specified project for this workspace is then updated. For this function, LASAL Class and LASAL Screen must be installed.
Create desktop shortcut	Generates a shortcut on the desktop of the current user for the selected workspace.
Expert Settings	Opens a drop-down menu, in which special, rarely used setting opportunities are offered. See Expert Settings
Help	Opens either the English or German help or opens an information box with version information.
Exit	Exits the LARSConfigTool.

4.2.1 Settings Wizard

Starts the configuration of the settings in a wizard.

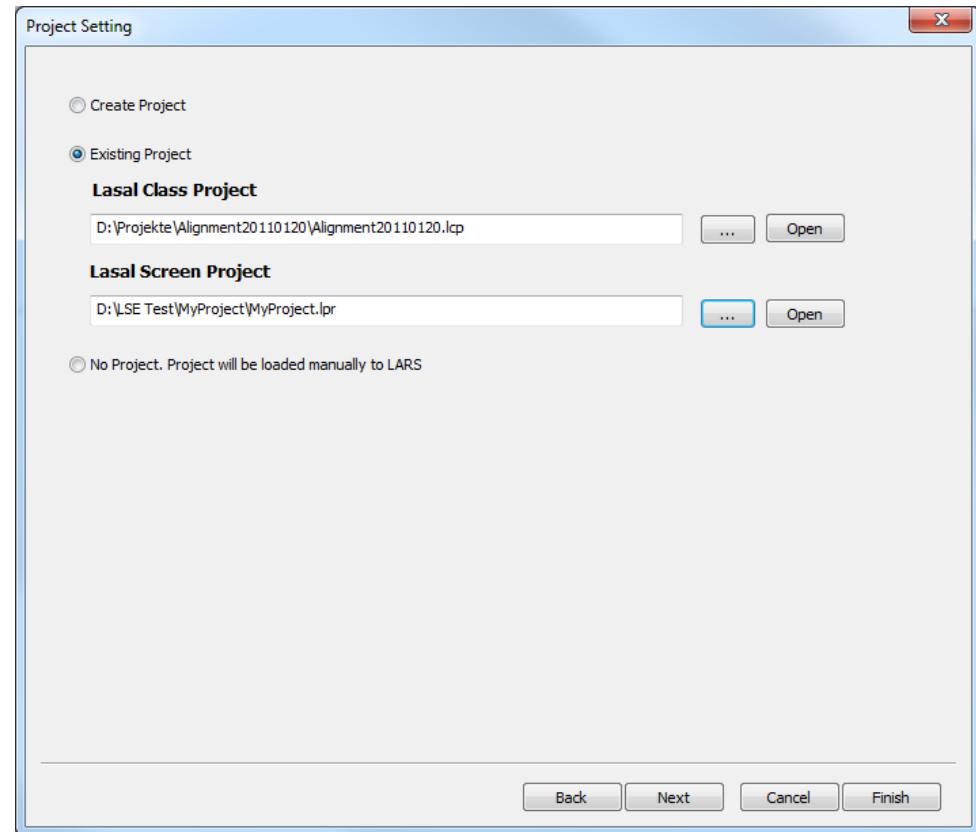
4.2.1.1 Path Setting



Lars Configuration name	The name of the workspace, must be unique. Umlauts are not allowed, just as numbers at the beginning.
Working Directory	C drive for LARS environment. Via the Search button, a path can be selected. If this C drive for the LARS environment is used at another configuration, a message appears:

Warning	
	The 'C:\LARS' working directory is already being used by 'DEFAULT' configuration
	<input type="button" value="OK"/>
New Folder	Here a new folder can be created in the currently selected folder.
Next	With the Next button, the next window of the wizard will be opened:
Cancel	With Cancel all changes will be reversed after confirmation.
Finish	With Finish all changes are taken (if, for example, only the name and the path should be changed, the following windows hereby can be skipped).

4.2.1.2 Project Setting



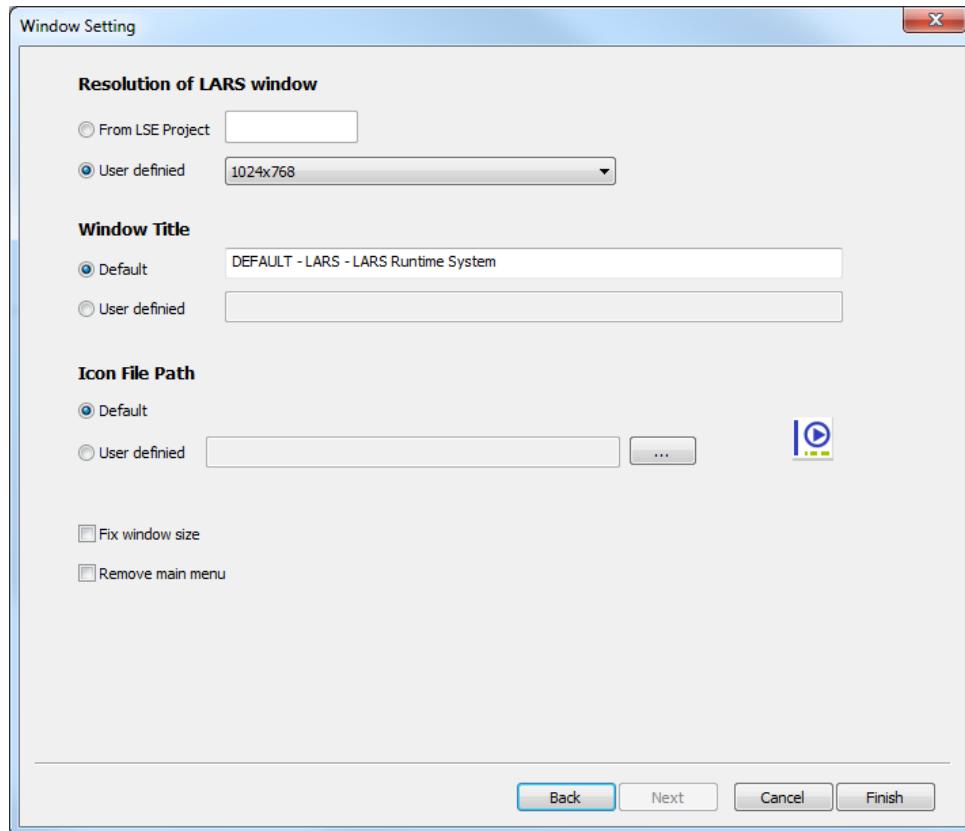
Create Project	Creates a new LASAL project.				
Existing Project	There are two possibilities here: <table border="1"><tr><td>Lasal Class Project</td><td>Path to a class project. Used to initialize the LARS environment or update the project. The path can be changed using the search button. With the More button, a path can be searched for and with the Open button LARS can be started and the project can be opened.</td></tr><tr><td>Lasal Screen Project</td><td>Path to a screen project. Used to initialize the LARS environment or update the project. The path can be changed using the search button. With the More</td></tr></table>	Lasal Class Project	Path to a class project. Used to initialize the LARS environment or update the project. The path can be changed using the search button. With the More button, a path can be searched for and with the Open button LARS can be started and the project can be opened.	Lasal Screen Project	Path to a screen project. Used to initialize the LARS environment or update the project. The path can be changed using the search button. With the More
Lasal Class Project	Path to a class project. Used to initialize the LARS environment or update the project. The path can be changed using the search button. With the More button, a path can be searched for and with the Open button LARS can be started and the project can be opened.				
Lasal Screen Project	Path to a screen project. Used to initialize the LARS environment or update the project. The path can be changed using the search button. With the More				

No Project. Project will be loaded manually to LARS	<p>button, a path can be searched for and with the Open button LARS can be started and the project can be opened.</p> <p>The workspace is created without a project. After its start a LASAL project must be manually loaded.</p>
----------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The buttons at the bottom of the screen have the same functionality as in the previous window. With Next the last Wizard window is opened:

4.2.1.3 Window Setting

Here the settings for the workspace window layout are made.



Resolution of LARS window	Setting for the LARS resolution. If a Screen project was set, the resolution can be determined automatically using "From LSE Project". With "User defined" pre-defined resolutions can be selected.
Window Title	Setting for an alternate window name. With "Default", the workspace name is used as title. Under "User Defined" any Unicode strings can be entered.
Icon File Path	With "Default" the preset icon path is used. Via "User Defined", an alternative icon path for LARS can be defined or selected one via the Search button.

Fix window size	When activated, the size of the LARS-window cannot be changed.
Remove main menu	When activated, the LARS menu bar is no longer displayed.
Finish	With the Finish button changes are stored.

4.2.2 Expert Settings

Opens a drop-down menu, in which special, rarely used setting opportunities are offered.

4.2.2.1 Reset Window Position

For each workspace, LARS remembers the position when ended. When changing from several to one screen, LARS may open in a non-visible area. In this case, the position of the LARS workspace must be reset to 0,0 with "Reset workspace position".

4.2.2.2 Edit Autoexec.lsl

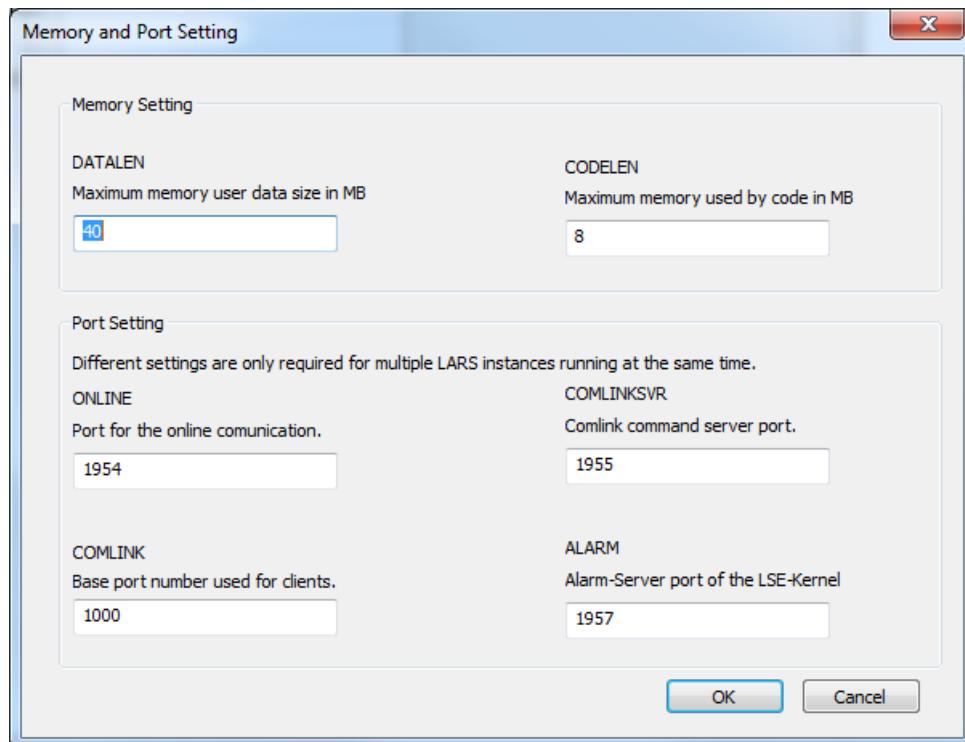
The init file Autoexec.lsl. is opened in an editor for direct editing.

4.2.2.3 Ping screen project server stations

If a Screen Project with remote stations is defined, these are "pinged" and the result is shown in a window:

4.2.2.4 LARS Memory and Port Setting

The Memory and Port Setting window is opened:



DATALEN Specifies the maximum user data memory.

CODELEN Indicates the maximum code memory.

The following settings must only be changed when multiple LARS instances are running at the same time. No setting changes are otherwise needed.

ONLINE TCP/IP port number of the online connection servers.

COMLINKSVR TCP/IP port number of the Comlink server (standard 1955). The Comlink-TCP/IP server requires two additional port numbers starting with the number defined in this parameter.

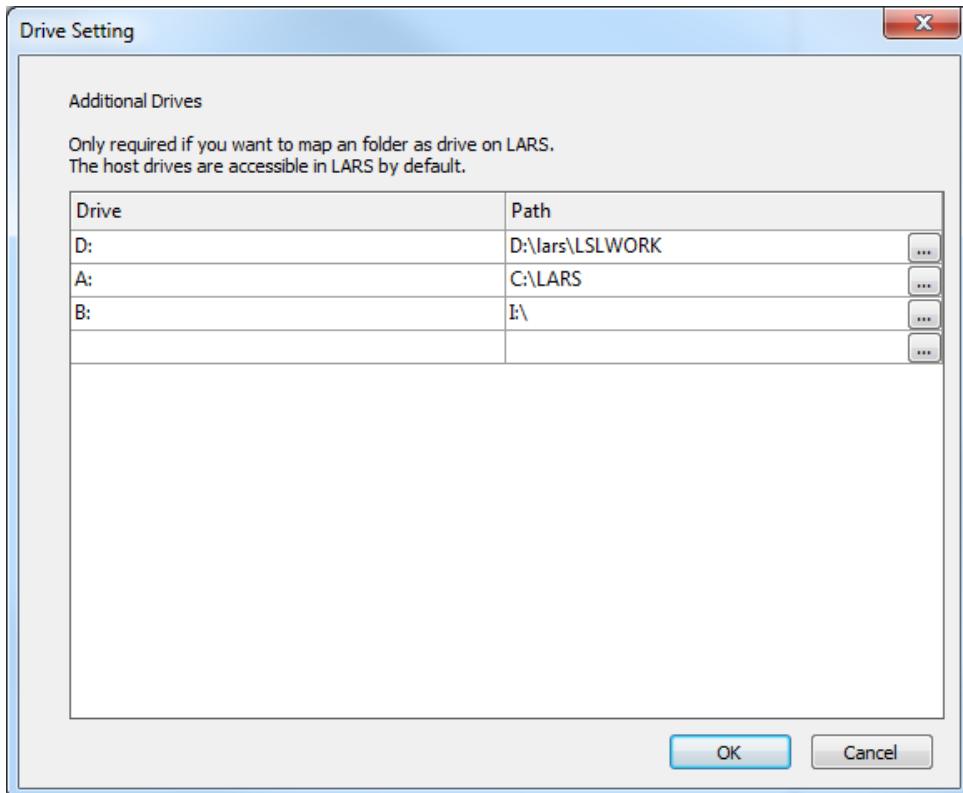
COMLINK TCP/IP port number of the internal program (default: 1000). This port number serves as the basis for the local port number of various system programs (e.g.: TCP/IP client in the alarm classes).

These system programs add their own offset to this base number, which is never greater than 10. This means that this base number for a LARS number is reserved for +9. If 1000 was entered for the first LARS instance, the next instance should have a base port number of 1010.

ALARM TCP/IP port number of the Alarm servers LSE kernel (default: 1957).

4.2.2.5 LARS Drive Mapping

Opens the Drive Setting dialog.

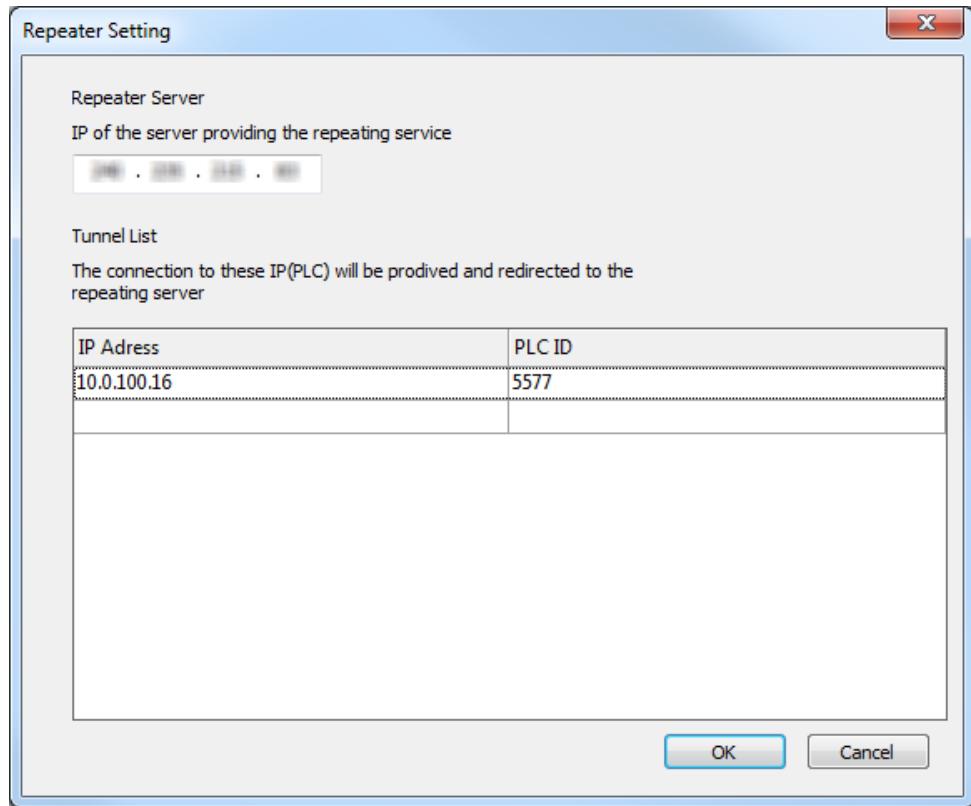


In this dialog, additional folders can be mapped as drives in LARS. The available drives in the host system can be accessed in LARS by default.

In the Drive column, drive letters can be selected. In the path column, the path can be selected via the Search button.

4.2.2.6 Repeater connection configuration

Opens the Repeater Setting dialog.



Working Directory Shows the C drive of the workspace.

4.3 Using LARS

After starting LARS with the LARS.EXE file, the CLI (command Line Interface) commands can be run or an online connection with LASAL Class or Screen can be made. A project can be downloaded, stored and tested.

In the Screen mode, it is possible to change between a window with a frame or full screen mode by pressing the keys ALT + ENTER or by selecting "View / Full screen" in the main menu.

4.3.1 Communication Interfaces

To go online with LASAL Class or Screen, the following interfaces are available:

- TCP/IP – The debug tool can either be run on a remote computer or on the same computer with LARS. To determine the IP address used by LARS, enter the command "IP" into the CLI.
- RS232
- CAN – LARS can only use the integrated CAN interface of an IPC. The CAN BUS interface BU 104 cannot be used, since the controller in this interface does not have all communication functions
LasalOS requires.

Communication between a visualization and a machine object:

- TCP/IP – This is the preferred interface for communication between a visualization and machine project.
- CAN – LARS can only use the integrated CAN interface of an IPC.

Communication interfaces, which use the APIs (Application Programming Interfaces):

- TCP/IP
- CAN – LARS can only use the integrated CAN interface of an IPC.

4.3.2 Programming Guidelines

Sensitive Instructions

The Intel architecture defines "privileged" and "sensitive" instructions. LARS runs LASAL applications on Privilege level 3 (CPL = 3), also called User mode, Ring 3 or I/O privilege level 0 (OPL = 0). LARS differs from LasalOS in this point. LasalOS runs a LASAL application on IOPL = 3.

The sensitive instructions (also called IOPL-sensitive) can only be run when $CPL \leq IOPL$ (I/O privilege level). Attempting to run a sensitive instruction when $CPL > IOPL$ generates a GP exception but does not cause a fatal error. LARS intercepts GP exceptions caused by running sensitive instructions and skips the affected command.

Sensitive instructions contain:

IN	Input
INS	Input string
OUT	Output
OUTS	Output string
CLI	Clear interrupt enable flag (IF)
STI	Set IF

Interrupts therefore cannot be disabled in LARS. Several applications try to disable interrupts to ensure that two tasks are not in the critical area at the time. A critical area is a section of the program that is not reentrant. For example, where global variables or hardware registers are accessed.

I/O instructions are special, since they are sensitive not only to IOPL, but also to the I/O Permission bitmap. The I/O permission bitmap is called, since in LARS, the CPL is larger than the IOPL. If the corresponding bits of an I/O Port are deleted, the I/O operation continues. Otherwise a GP exception is generated, which is intercepted and the instruction is skipped. In the current LARS version (starting from version 5.44) all bits in the I/O Permission bitmap do not = null. This means that the output commands have no effect and the input commands have an undefined result.

For LARS to function, this limit must be considered when writing a new application or porting an existing one.

Interval for Cyclic Functions (CyWork RtWork or Background)

In LARS, the smallest interval cyclic functions differs from the smallest interval of a LASAL operating system. In the current LARS version (starting from version 5.44) the values are as follows:

- Real time 5 ms
- Cyclic, Background: 10 ms

4.3.3 Printing with LARS

There are two options for using a Windows printer with LARS:

1. With the menu:

This option prints the actual visible screen on the selected printer (pop-up dialog).

2. Using the printer classes from the application:

This option prints a specified screen (not necessarily the visible screen) on the selected printer (pop-up dialog). The application task, which triggers the print job, is paused until the print job is complete.

5 Communication and Driver

5.1 Online Communication

The online communication is used for data exchange between an application program on a PC (e.g. debug tool) and the LASAL operating system. Direct communication with objects in a LASAL project is not possible.

The PC program sends a command to the LASAL operating system then waits for an answer. The LASAL operating system cannot send data, only an acknowledgment for a command.

Online communication in the PC is implemented in the LASAL32.dll, which exports a package of functions that can be called by a Windows program. These functions are called LASAL32 API functions.

The LASAL32 API functions can be organized under the following categories:

- Online connection administration (establish or remove a connection)
- Modem functions (dial, hang-up)
- Data exchange (receive data, send data)
- Debug functions (CPU status inquiry, breakpoints, register contents, start/stop)
- Software update (operating system update, module download, progmemo)
- File transfer
- Operating system functions (CPU load, trace-records, system log)

Implementation in the operating system

A separate task is started for each online connection. The priority of these tasks is the same as those for cyclic tasks. If there is no communication over a specified time, the communication task stops automatically.

5.2 DebugIP Communication

DebugIP communication allows an application program on a PC (e.g. debug tool) to communicate with a command interpreter in the LASAL project. On the PC, the debugIP communication is implemented in both the LslOnline.dll and the LASAL Class code directly.

The following functions are available for the application program:

- Establishing and terminating a connection
- Accessing the address of an object using the object name.

- Accessing the class name of an object using the object address.
- Call the write-method of a server.
- Call the read-method of a server.

Details

Communication buffers are used for communicating with the command interpreter, whereby every debug tool has its own communication buffer. The individual fields in the communication buffer are described and read by the functions in the LASAL32.dll (SetData, GetData).

A cyclic program runs in the control loader, which checks all active communication buffers as to whether it is necessary to execute command.

So that a debug tool is assigned a communication buffer, an I_REGISTER command has to be executed using a standard communication buffer. The standard communication buffer is protected with a lock mechanism to prohibit multiple debug tools from using the buffer at the same time. If a communication buffer is inactive for a specified time period, it is then removed from the loader.

The Communication buffer connects with the command interpreter as follows:

- The communication buffers WorkState field is defined state by writing WS-QUIT.
- The command is sent.
- The WorkState field is then set to WS_BUSY. As soon as the command interpreter finds WS_BUSY in the PLC, the command is executed.
- The WorkState field is queried until a result is returned.
- The length of the result is read.
- The result is read.

5.3 Comlink

Data between visualizations (client) and controls (server) are exchanged via the Comlink. The client either sends a command to the server and waits for the answer or puts a so-called Update-Cell into the server's update list. An Update-cell in the update list instructs the server to monitor a variable and to inform the client when the value changes.

The server contains 2 update lists; one is static and one dynamic. The client decides if an Update-cell should be inserted into the static or dynamic update list. Normally the static update list is initialized when the project is started and remains unchanged, while the dynamic update-list is updated if new values must be entered in the variable (e.g. if a display with new variables is selected).

The Comlink interface contains the following functions:

- Login – reserves a communication channel and establishes a connection to a local or remote station.
- TxCommand – sends a command to the command interpreter and waits for an answer.
- TxUpd – transfers the UpdateCell to an update list.
- StartStopRefresh – informs the server of the number of UpdateCells in the update list that should be scanned.
- InstallCallback – installs a callback function that is called, as soon as the value of an UpdateCell changes.

Communication Channel

Data are exchanged using a communication channel (commands, answers and changes in the update list).

The first channel is designed for local communication. 5 predefined communication channels are available for the CAN. For the TCP/IP, 16 communication channels are available and are allocated as required.

Application flow at login

When the login function is called, the application reserves a communication channel and establishes a connection to the local or remote station.

CAN

The connection is established using the so-called Channel16, whereas a Channel16 command is sent to the target operating system over the station object number 0x700 + target station. The channel16 command contains the desired Comlink object number of the outstation. The Comlink object number is the outstation number +1. After receiving the Chanel16 command, the outstation's operating system writes the object number contained to a system variable (OPS.CH16buf). The loader then queries this variable. If its contents of the variable are valid, it is acknowledged and the connection is established.

TCP/IP

A connection is established to the outstation port 1955 (command port) and 1956 (refresh port). Commands are sent and returns received over the command connection. Changes in the update list are received over the refresh connection.

RS232

Currently not used.

PC: PLC:

6 Online Communication

6.1 Communication Driver Lasal32 API

6.1.1 Overview

The Lasal32 application programming interface (API) supports the development of applications, which are run on an operating system for the communication with the PLC (runs on a LASAL operating system (LasalOS)).

6.1.2 Lasal32 API for Windows

The Lasal32 API does not function with an operating system that is older than Windows 98 or Windows NT 4.0 (such as Windows 95, Windows 3.x or Windows NT 3.xx). The API is a Dynamic Link Library (DLL) and is implemented by series of Windows drivers.

The Lasal32 API consists of the following files:

- Lasal32.dll (Dynamic Link Library)
- Lasal32.lib (library with the exported DLL functions)
- Lasal32.h (C header file)

6.1.3 Lasal32 API for Linux

The Lasal32 API consists of the following files:

- libLasal32.so (Linux Shared Object)
- Lasal32.h (C header file)

For Linux, only TCP communication is supported.



6.1.4 Lasal32 API Functions

The documentation for the Lasal32 API functions is divided into the following categories:

- Managing the online connection
- Modem functions
- Data exchange

- Status information and PLC instructions
- Administrative functions
- File loading

Lasal32.dll is thread-safe starting with version 1.34. They are protected with a critical section. It is important to note that in earlier versions, the application is responsible for protecting the functions from repeated calls in multi-thread applications.

6.1.5 Error Handling

Typically, most Lasal32 API functions return an output an error when a value of Null (FALSE) is returned. With an error, the LASAL32 API functions call SetLastError to set the last error code for the active thread.

Applications can call the values stored by this function with the GetLastError function to find the cause of a function error. The last error code is stored in the local thread memory, in order to prevent that multiple threads overwrite the values of each other.

The following table contains a list of the error codes. They are returned by the GetLastError function when several of the Lasal32 API functions have an error.

Code	Description	Name
0x20000001	Access to the interface is denied.	ERROR_SIGMA32_ACCESSDENIED
0x20000002	No connection was opened when Online was called.	ERROR_SIGMA32_NOTOPEN
0x20000003	Online was called before the connection with the Offline call was completed.	ERROR_SIGMA32_ALREADYOPEN
0x20000004	The function does not support the interface.	ERROR_SIGMA32_NOTDEFINEDFORV24
0x20000005	The parameter is incorrect.	ERROR_SIGMA32_WRONGPARAMETER
0x20000006	A write timeout is triggered.	ERROR_SIGMA32_WRITE_TIMEOUT
0x20000007	A read timeout is triggered	ERROR_SIGMA32_READ_TIMEOUT
0x20000008	A send process has malfunctioned.	ERROR_SIGMA32_WRITE_FAULT
0x20000009	A read process has malfunctioned.	ERROR_SIGMA32_READ_FAULT

0x2000000C	An existing connection was made.	ERROR_SIGMA32_CONN_RESET
0x2000000D	A callback function (CB_FUNCTYPE or CB_FUNCTYPE2) has aborted a data transfer with the return value CBRETURN_ABORT.	ERROR_SIGMA32_ABORTED_BY_USER
0x2000000E	The remote PLC shows an error message.	ERROR_SIGMA32_REMOTE_ERROR
0x2000000F	A function was called that is not supported by the remote PLC.	ERROR_SIGMA32_UNSUPPORTED_CMD
0x20000010	An unknown general error has occurred.	ERROR_SIGMA32_GENERAL_ERROR
0x20000011	A connection could not be established.	ERROR_SIGMA32_CONNECT_FAILED
0x20000012	Unallowed function.	ERROR_SIGMA32_INVALID_FUNCTION
0x20000013	Timeout waiting for a specific CPU status.	ERROR_SIGMA32_CPUSTATUS_TIMEOUT
0x20000014L	No memory available.	ERROR_SIGMA32_OUT_OF_MEM
0x20000015L	Max. number of connections reached.	ERROR_SIGMA32_MAX_CONN
0x20000016L	Invalid ocb number	ERROR_SIGMA32_INV_OCB_NBR
0x20000017L	Invalid project status	ERROR_SIGMA32_INV_PRJ_STATE
0x20000018L	Buffer is too small	ERROR_SIGMA32_OUT_OF_BUFSIZE
0x20000019L	Buffer is too small	ERROR_SIGMA32_INVALID_DATA
0x2000001AL	Incorrect RLB (RefreshListBlock)	ERROR_SIGMA32_INVALID_RLB
0x2000001BL	The maximum length was exceeded	ERROR_SIGMA32_LENGTH_TO_BIG
0x2000001CL	Incorrect variable type	ERROR_SIGMA32_WRONG_VAR_TYPE
0x2000001DL	Incorrect length	ERROR_SIGMA32_LENGTH_WRONG

0x200000 38L	Wrong online password	ERROR_SIGMA32_WRONG_ONLINE_PW D
0x200000 40L	PLC has no online password	ERROR_SIGMA32_NO_PASSWORD_SET

6.1.6 Managing the Online Connection

6.1.6.1 IsOnline

Corresponds to [IsOnlineH\(\)](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS IsOnline(void);
```

6.1.6.2 IsOnlineH

The IsOnlineH() function checks an existing online connection to the PLC.

```
extern "C" LSL_BOOL LASAL32_EXPORTS IsOnlineH(int32_t ocbNum);
```

Transfer parameters	Type	Description
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())
Return parameters	Type	Description
		<p>TRUE Success; connection is active</p> <p>FALSE Error</p> <p>To query extended error information, call the GetLastError() function.</p>

6.1.6.3 LslPing

Sends a ping signal to a remote host and checks the answer.

```
extern "C" int32_t LASAL32_EXPORTS LslPing(
const char* host,
DWORD dwReadTimeout,
SLs1PingInfo* pInfo
);
```

Transfer parameters	Type	Description
host	const char*	IP-Adresse des Remote-Host als ASCII-String
dwReadTimeout	DWORD	Timeout time until a response is received in [ms]

plInfo	SLsIPingInfo*	Pointer to a variable of type SLsIPingInfo. Before calling the function, the size of the structure must be entered in the dwSize field.				
Return parameters	Type	Description				
		<table border="1"> <tr> <td>0</td><td>Success</td></tr> <tr> <td><0</td><td>Error or timeout</td></tr> </table>	0	Success	<0	Error or timeout
0	Success					
<0	Error or 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

Example

```
SLsIPingInfo pingInfo;
memset(&pingInfo, 0, sizeof(pingInfo));
pingInfo.dwSize = sizeof(pingInfo);
int nRes = LsIPing("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

Releases the resources, which were assigned/occupied with [OcbOpen\(\)](#).

```
extern "C" extern "C" void LASAL32_EXPORTS OcbClose(
int32_t ocbNum
);
```

Transfer parameters	Type	Description

ocbNum	int32_t	Online Connection Block number (return value of OcbOpen)
--------	---------	---------------------------------------------------------------------------

6.1.6.5 OcbOpen

Opens the Online Control Block and returns the handle. A maximum of 64 handles are possible; in the event of an error, -1 is returned.

Several online connections per DLL instance are possible with these OCBs. If the OCB is no longer required, it must be released again.

```
extern "C" int32_t WINAPI OcbOpen();
```

6.1.6.6 Offline

Corresponds to [OfflineH\(\)](#) with ocbNum = 0.

```
extern "C" void LASAL32_EXPORTS Offline(void);
```

6.1.6.7 OfflineH

The OfflineH() function deactivates an online connection with the connection entered in ocbNum.

Even if Online() fails, Offline() must be called afterwards.



```
extern "C" void LASAL32_EXPORTS OfflineH(int32_t ocbNum);
```

Transfer parameters	Type	Description
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen)

6.1.6.8 Online

The Online() function establishes a connection to a remote PLC.

```
extern "C" LSL_BOOL LASAL32_EXPORTS Online(
    const char *szComm,
    BYTE uBaudRate,
    BYTE uPGStation,
    BYTE uPLCStation,
    BYTE uAutoInit
);
```

Transfer parameters	Type	Description																		
SzComm	const char *	<p>Pointer to a null-terminated string, which indicates the interface that is used to establish the connection.</p> <table border="1"> <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>Before a connection is created via the Modem, an active Modem connection must be available. See also: Functions for managing a Modem connection.</td></tr> <tr><td>TCP/IP</td><td>TCP:<IP Address>[:<Port>];ROUTE=<Route-Nr> <IP-Adresse> defines the IP-Adresse in the standard Internet "." Notation (dotted). <Port> defines the port number used by the remote PLC. Port is optional, the default value is 1954. If you want to go online via a route, this can be done via the optional parameter ROUTE. <Route-Nr> is the route number assigned on the PLC.</td></tr> <tr><td>LPT1</td><td>CAN bus adapter BU104 at LPT1</td></tr> <tr><td>LPT2</td><td>CAN bus adapter BU104 at LPT2</td></tr> <tr><td>USBCAN</td><td>CAN bus adapter BU106 at USB</td></tr> </table> <p>Here, additional options can be defined in szComm.</p> <p>TCP_NODELAY=x x=0 same as TCP_NODELAY_OFF x=1 same as TCP_NODELAY_ON</p> <p>TCP_NODELAY_ON Disables the Nagle algorithm for sending coalescence</p> <p>TCP_NODELAY_OFF Enables the Nagel algorithm for sending coalescence</p> <p>TCP_TIMEOUT=x The same as TCP_RD_TIMEOUT=x and TCP_WR_TIMEOUT=x</p> <p>TCP_RD_TIMEOUT=x Receive timeout in seconds</p> <p>TCP_WR_TIMEOUT=x Send timeout in seconds</p> <p>PING_TIMEOUT=x</p>	COM1	X3 – X4	COM2	X3 – X4	COM3	X3 – X4	COM4	X3 – X4	COMM	Before a connection is created via the Modem, an active Modem connection must be available. See also: Functions for managing a Modem connection.	TCP/IP	TCP:<IP Address>[:<Port>];ROUTE=<Route-Nr> <IP-Adresse> defines the IP-Adresse in the standard Internet "." Notation (dotted). <Port> defines the port number used by the remote PLC. Port is optional, the default value is 1954. If you want to go online via a route, this can be done via the optional parameter ROUTE. <Route-Nr> is the route number assigned on the PLC.	LPT1	CAN bus adapter BU104 at LPT1	LPT2	CAN bus adapter BU104 at LPT2	USBCAN	CAN bus adapter BU106 at USB
COM1	X3 – X4																			
COM2	X3 – X4																			
COM3	X3 – X4																			
COM4	X3 – X4																			
COMM	Before a connection is created via the Modem, an active Modem connection must be available. See also: Functions for managing a Modem connection.																			
TCP/IP	TCP:<IP Address>[:<Port>];ROUTE=<Route-Nr> <IP-Adresse> defines the IP-Adresse in the standard Internet "." Notation (dotted). <Port> defines the port number used by the remote PLC. Port is optional, the default value is 1954. If you want to go online via a route, this can be done via the optional parameter ROUTE. <Route-Nr> is the route number assigned on the PLC.																			
LPT1	CAN bus adapter BU104 at LPT1																			
LPT2	CAN bus adapter BU104 at LPT2																			
USBCAN	CAN bus adapter BU106 at USB																			

		<p>Time (rtt - round trip time) until a response is received in milliseconds PING_AMOUNT=x</p> <p>Number of pings sent when establishing a connection CONNECT_TIMEOUT=x</p> <p>Timeout for TCP Connect in milliseconds ONLINE_TIMEOUT=x</p> <p>Maximum age of the last successful communication in milliseconds, from which a new determination is made as to whether the online connection still exists TCP_NOT_IN_LAN</p> <p>The PLC is not in the LAN. Therefore no ping when establishing a connection and a long timeout for the TCP connect.</p>
uBaudRate	BYTE	<p>Indicates the baud rate, with which the RS232 or CAN bus works or a read timeout of a TCP / IP connection.</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 • 250 kb • 125 kb • 100 kb • 50 kb • 20 kb <p>This parameter defines the timeout of a read function for a TCP / IP connection in seconds. A value of 0 means that the default read timeout should be used.</p> <p>This parameter has no meaning for a Modem connection and should be set to 0.</p>
uPGStation	BYTE	This parameter provides the number of CAN stations to the PC (0-31). This parameter is invalid for other connection types and should be set to 0.
uPLCStation	BYTE	This parameter provides the number of CAN stations to the PLC (0-31). This parameter is invalid for other connection types and should

		be set to 0.
Uautolnit	BYTE	This parameter is obsolete and should be set to 0
Return parameters	Type	Description
		TRUE Success
		FALSE Error
		To query extended error information, call the GetLastError() function.

Use [Offline\(\)](#) to terminate the connection. An application should always have a call to Offline for each Online() call, even if Online() fails, in order to return connection resources to the system. [IsOnline\(\)](#) is used to check whether a connection still exists after a function that requires one fails. To correct such an error, use [Offline\(\)](#) and try to reestablish the connection.

If a TCP/IP connection is used, the remote PLC terminates the connection automatically when no data is sent within 30 seconds (newer operating systems use a timeout value of 60 seconds). To establish a TCP/IP connection, all functions that require a connection (e.g. [GetCpuStatus\(\)](#)) must be called in regular intervals.

Example

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: Default
          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

Corresponds to the [Online\(\)](#) function with the difference that here an additional parameter ocbNum (Online Control Block) can be defined. OcbNum indicates which of the 64 possible online control blocks should be used to establish the connections; this number is retrieved with the [OcbOpen\(\)](#) command.

```
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

Corresponds to [OnlineOptionsH\(\)](#) with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS OnlineOptions(
void      *pData,
uint32_t  dOption
);
```

6.1.6.11 OnlineOptionsH

Set optional functionalities when establishing the connection.

```
extern "C" LSL_BOOL LASAL32_EXPORTS OnlineOptions(
int32_t      ocbNum,
void      *pData,
uint32_t  dOption
);
```

Transfer parameters	Type	Description
---------------------	------	-------------

ocbNum	int32_t	Online connection block number (return value of OcbOpen())				
pData	*pData	Pointer to the data				
dOption	uint32_t	Option to be set. Available are: LSL_ONLINE_TCP_CANCEL_CONNECT_FLAG: sets an abort condition for the TCP_Connect command. The pointer pData is entered in the TCP control block. pData must point to a Boolean variable. This Boolean variable is queried cyclically during the TCP connection setup. As soon as it contains TRUE, the connection is aborted.				
Return parameters	Type	Description				
		<table border="1" style="margin-left: auto; margin-right: auto;"> <tr> <td>TRUE</td><td>Success</td></tr> <tr> <td>FALSE</td><td>error</td></tr> </table> <p>To query extended error information, call the GetLastError() function.</p>	TRUE	Success	FALSE	error
TRUE	Success					
FALSE	error					

Example

```
BOOL rc = FALSE;
BOOL CancelTCPConnect = FALSE;
rc = OnlineOptionsH(ocbNum, &CancelTCPConnect, LSL_ONLINE_TCP_CANCEL_CONNECT_FLAG);
```

6.1.6.12 OnlinePwd

Corresponds to the [OnlineH\(\)](#) function with the additional onlPwd parameter. When onlPwd = NULL, the function is identical. The password test is only performed with a TCP/IP connection. The password is entered into the function in plaintext.

```
extern "C" LSL_BOOL LASAL32_EXPORTS OnlinePwd(
    const char    *szComm,
    BYTE          uBaudRate,
    BYTE          uPGStation,
    BYTE          uPLCStation,
    BYTE          uAutoInit,
    const char    *onlPwd
);
```

If the password is incorrect, FALSE is returned and [GetLastError\(\)](#) returns the error code ERROR_SIGMA32_WRONG_ONLINE_PWD.

If you try to go online with a password, but the PLC is not password protected, the OnlinePwd() function will succeed, but no connection will be established and [GetLastError\(\)](#) will return the error code ERROR_SIGMA32_NO_PASSWORD_SET. [IsOnline\(\)](#) can be used to check whether an online connection exists.

Example

```

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

Corresponds to the [OnlineH\(\)](#) function expanded with the password function described above.

```

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

Corresponds to OnlineSSLH() with 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

Establishes an online connection between the PC (project) and the PLC.

```

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
);

```

Transfer parameters	Type	Description
ocbNum	int32_t	Online connection block number (return value of OcbOpen())
szComm	const char*	Pointer to a 0-terminated string that specifies the interface used to establish the connection. Important: port 2054 must be specified for an SSL connection. E.g.: "TCP:10.10.170.190:2054". See also OnlineH() .
uBaudRate	uint8_t	Specifies the baud rate at which the RS232 or CAN BUS interface operates or a read timeout for a TCP/IP connection. Details see OnlineH() .
uPcStation	uint8_t	Number of CAN stations to the PC (0-31). This parameter is invalid for other connection types and should be set to 0.
uSpsStation	uint8_t	Number of CAN stations of the PLC (0-31). This parameter is invalid for other connection types and should be set to 0.
uAutoInit	uint8_t	Parameter is obsolete and should be set to 0.
onlPwd	const char*	Pointer to the password for the online connection. This parameter must be NULL if the PLC does not have an online password. Details see OnlinePwdH() .
bSSL_TLS	const bool	TRUE: SSL/TLS should be used
pSSL_TLS_Callback	SSL_VerifyServerCert_FUNCTYPE	Pointer to a callback function of the user to check the server's certificate. This parameter may be NULL if bSSL_TLS==FALSE.
Return parameters	Type	Description

		TRUE Success
		FALSE error
To query extended error information, call the GetLastError() function.		

CallBack Function Prototype

```
LSL_BOOL _cdecl SSL_VerifyServerCert _FUNCTYPE(void*, uint32_t);
```

Transfer parameters	Type	Description
	void*	Pointer to SSvrCertInfo with the server certificate
	uint32_t	Size of SSvrCertInfo
Return parameters	Type	Description
		TRUE If the user trusts the certificate FALSE If the user does not trust the certificate

Example

```
char onlPwd[32] = { 0 };
const char* pPwd = onlPwd;

strcpy(onlPwd, "OnlinePassword");

if (OnlineSSL("TCP:10.10.170.190:2054", 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 Modem Functions

The modem functions can be used to create an active modem connection. To use these functions, the modem must first be installed in the operating system with the correct parameters. More information can be found in the Modem Connection in LASAL documentation.

6.1.7.1 ModemOpen

The ModemOpen() function installs the Modem module. The modem module must be initialized before the rest of the modem functions can be called.

```
extern "C" LSL_BOOL LASAL32_EXPORTS ModemOpen(void);
```

Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error

6.1.7.2 ModemGetNumber

The ModemGetNumber() function returns the number of installed modems.

```
extern "C" DWORD LASAL32_EXPORTS ModemGetNumber(void);
```

Return parameters	Type	Description	
		Number of available modems	

6.1.7.3 ModemClose

The ModemClose() function deactivates an active connection.

```
extern "C" void WINAPI ModemClose(void);
```



After the connection was deactivated, the modem module must be reinitialized with [ModemOpen\(\)](#) before the remaining modem functions can be called.

Example

```
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

The ModemCall() function starts the selection process.

```

extern "C" LSL_BOOL LASAL32_EXPORTS ModemCall(
    DWORD    dModemID,
    BYTE     uType,
    DWORD    dConfigBaud,
    char    *szTelNr,
    BYTE     nTimeout
);

```

Transfer parameters		Type	Description	
dModemID	DWORD	A 0-based index of the modem		
uType	BYTE	Specifies the connection type. This parameter must be set to 1 (= direct connection type), since no other connection types are used at this location.		
dConfigBaud	DWORD	Indicates whether pulse or tone selection is used		
		0 Tone selection 1 Pulse selection		
szTelNr	CHAR	Pointer to a null-terminated string, which contains the telephone number		
nTimeout	BYTE	Timeout, value in seconds		
Return parameters		Type	Description	
			TRUE Success FALSE Error	

6.1.7.5 ModemGetName

The ModemGetName() function returns a description of a specific modem.

```
extern "C" LSL_BOOL LASAL32_EXPORTS ModemGetName(
    DWORD    dModemID,
    char     *pszName,
    WORD     maxsize
);
```

Transfer parameters	Type	Description	
dModemID	DWORD	A null-based index of the modem	
pszName	CHAR	Pointer to a character buffer, which contains the modem description	
maxsize	WORD	Defines the size of the character buffer pszname	
Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error

6.1.8 Data Exchange Functions

Data Exchange with Servers

LslGetObject() or LslGetObjectEx() returns the channel address of the server.

LslReadFromSvr() and LslWriteToSvr() can be used to access a numerical server.

A string server can be accessed with LslReadFromSvrStr() and LslWriteToSvrStr().

Data Exchange with Clients

LslGetObject() or LslGetObjectEx() returns the channel address of the client.

The client can be accessed with LslReadFromClt() and LslWriteToClt().

Data Exchange with Global Variables

LslGetAdressVar() returns the address of a global variable.

The global variable can then be accessed with GetData(), LslGetDataEx() and SetData().

Data Exchange with Global Variables

Data Exchange with a Memory Address on the PLC

Object member variables of a class or memory areas allocated with malloc() cannot be accessed without further ado because the addresses cannot be determined. This problem could be solved as follows.

A request server and an address server are implemented in the relevant class. The address to be displayed on the address server is entered in the request server.

If only one or a few addresses need to be accessed, the request server can be dispensed with. Only one address server or a small number of address servers are then implemented, which permanently display the desired addresses. The displayed address can be read with LslGetObject("Objektname.AdressenServername", ...) and LslReadFromSvr(). The determined address can then be accessed with GetData(), LslGetDataEx() and SetData(), LslSetDataEx().

Address of an Array Element

The address is calculated as the array base address + index * sizeof(array element).

Address of a Structure Element

The address is calculated as the structure base address + offsetof(StructType, Element).

Attention: the address is only correct if the alignment on the PC and on the PLC is the same.

Exchange of Large Amounts of Data

The functions LslGetDataEx(), LslGetDataListEx(), LslGetDataListSpecial() and GetDataList() together with SendDataList() are suitable for reading large amounts of data.

The LslSetDataEx() function is suitable for writing large amounts of data.

6.1.8.1 GetData

Corresponds to [GetDataH\(\)](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetData(
    void        *pBuffer,
    uint32_t    addr0,
    uint16_t    nCount
);
```

6.1.8.2 GetDataH

The GetDataH() function calls data from the PLC.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetDataH(
    int32_t    ocbNum,
    void       *pBuffer,
    uint32_t   addr0,
    uint16_t   nCount
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())	
pBuffer	void	Pointer to the memory area in which the received data should be stored	
addr0	uint32_t	Address of the data area in the PLC	
nCount	uint16_t	Number of records bytes to receive	
Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error
		To query extended error information, call the GetLastError() function.	

Example

```
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

Corresponds to [GetDataListH\(\)](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetDataList(
    char* data0
);
```

6.1.8.4 **GetDataListH**

The `GetDataListH()` function calls data, whose addresses were entered in a previous call of `SendDataListH()`.

The `LslGetObject()` and `LslGetObjectEx()` functions can be used to determine the addresses of class objects and of clients and servers.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetDataListH(
    int32_t    ocbNum,
    char       *data0
);
```

Transfer parameters		Type	Description											
ocbNum		int32_t	Online Connection Block number (return value of OcbOpen())											
data0		char	Pointer to a buffer that calls the data list. The data list has the following format: <table border="1" data-bbox="464 626 1021 849"> <thead> <tr> <th>Offset</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Unsigned short</td> <td>Number of addresses</td> </tr> <tr> <td>2</td> <td>Unsigned char</td> <td> Data content: Content of the 1st address, Content of the 2nd address, and so on </td> </tr> </tbody> </table> The address and length are not in the data list. The application is responsible for assigning a buffer that is large enough to receive the data, whose address range was defined in a call to <code>SetDataList()</code> .			Offset	Type	Description	0	Unsigned short	Number of addresses	2	Unsigned char	Data content: Content of the 1 st address, Content of the 2 nd address, and so on
Offset	Type	Description												
0	Unsigned short	Number of addresses												
2	Unsigned char	Data content: Content of the 1 st address, Content of the 2 nd address, and so on												
Return parameters														
		Type	Description											
			TRUE	Success										
			FALSE	Error										
To query extended error information, call the GetLastError() function.														

Example

```
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 !\n" );
    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

Runs the Kill method of the specified server.

```

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
);

```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())	
dwSrvAddress	uint32_t	Pointer to the server	
pResult	uint8_t	Pointer to the result buffer	
dwResultCnt	uint32_t	Size of the result buffer	
useLenField	uint32_t	Number of event data in the length section of the parameter	
returnState	uint32_t	Set to TRUE, in order to query the status of the method called kill	
Return parameters	Type	Description	
		TRUE	Success

		FALSE Error
--	--	-------------

Example

```
//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 = LslExecKill( addr, (unsigned char* )&retbuf[ 0 ],
256, FALSE, TRUE );
if( ret )
{
m_output += "Killed\r\n";
KillTimer( 2 );
}
else
m_output += "Kill failed\r\n";
```

The first call from LslExecNewInstrEx() calls the NewInst method of the specified server. Depending on the status of the NewInst method, subsequent calls are made for either NewInst (if the previous call has been completed and the function is set to READY) or GetState (If the previous call is still being processed and the function is set to BUSY). A busy NewInst ends automatically when the connection is interrupted or no further status queries are made. To end a BUSY NewInst, the user can call the LslExecKill function. This call ends the server method and resets the status to READY. Calling a single NewInst method several times is not supported (subsequent calls will trigger the GetState function).

6.1.8.6 LslExecNewInstrEx

Corresponds to the [LslExecNewInstrExH\(\)](#) function with 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

Runs the NewInst/GetState methods of the specified server.

```
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
);

```

Transfer parameters	Type	Description				
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())				
dwSrvAddress	DWORD	Pointer to the server				
wCmd	WORD	Command for running by the NewInst/GetState method				
adwParas[]	DWORD	Array of method parameters (4 bytes per parameter)				
dwParaCnr	DWORD	Number of parameters in the above array. A maximum of 20 parameters is possible.				
pResult	UNSIGNED CHAR	Pointer to the result buffer or NIL if no return data is required. The first WORD in the buffer contains the length of the data including this length field. The maximum length is 252 bytes.				
dwResultCnt	DWORD	Number of bytes expected from the response (useLenField = 0) or size of the result buffer (useLenField ≠ 0).				
useLenField	DWORD	0: At least dwResultCnt bytes are expected from the response. If less data is available, the function fails. If more data is available, the superfluous data is discarded. #0: An attempt is made to pass the entire result buffer to the caller. This means that the length field in the response is interpreted and only the number of bytes defined in it is copied to the result buffer. dwResultCnt is the size of the caller's buffer. If dwResultCnt is too small, the function fails.				
returnState	DWORD	Set to TRUE, in order to query the called NewInst/GetState methods. The status is returned in the first WORD of the result buffer.				
Return parameters	Type	Description				
		<table border="1"> <tr> <td>TRUE</td><td>Success</td></tr> <tr> <td>FALSE</td><td>Error</td></tr> </table>	TRUE	Success	FALSE	Error
TRUE	Success					
FALSE	Error					

With the instruction call, objects commands can be triggered. The commands are run in the NewInst method of the class. Instruction calls are often used for data transfer.

6.1.8.8 LslGetAdressVar

Corresponds to [LslGetAdressVarH\(\)](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetAdressVar(
    const char    *name,
    int32_t        *adresse
);
```

6.1.8.9 LslGetAdressVarH

The LslGetAdressVarH() function specifies the address of a global variable in the PLC.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetAdressVarH(
    int32_t        ocbNum,
    const char    *name,
    int32_t        *adresse
);
```

This function cannot determine the address of a class object, function or element in a structure.



Transfer parameters	Type	Description	
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())	
name	CONST CHAR	Pointer to a 0-terminated string that contains the name of the variable	
Address	int32_t	Defines a pointer to a variable, which contains the address of the global variable. If the parameter contains a value of 0, the address of the global variable cannot be determined.	
Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error
		To query extended error information, call the GetLastError() function.	

Example

```
unsigned long address;
char *name = "OPS";

if ((LslGetAdressVar( 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

The function corresponds to [LslGetDataExH\(\)](#) with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDataEx(
    uint8_t    *pBuffer,
    uint32_t   addr0,
    uint32_t   nCount
);

```

6.1.8.11 LslGetDataExH

The LslGetDataExH() function calls data from the PLC (over 32000 bytes can be read)

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDataExH(
    int32_t    ocbNum,
    uint8_t    *pBuffer,
    uint32_t   addr0,
    uint32_t   nCount
);

```

Transfer parameters		Type	Description	
ocbNum		int32_t	Online Connection Block number (return value of OcbOpen())	
pBuffer		uint8_t	Pointer to the memory area to which the received data should be copied	
addr0		uint32_t	Address of the data in the target system	
nCount		uint32_t	Number of records bytes to receive	
Return parameters		Type	Description	
			TRUE	Success
			FALSE	Error
With LslGetError, the last error code can be determined. This code corresponds either to a Lasal32 or system specific error code (Windows: GetLastError() or Linux: errno).				

Example

```

BYTE pBuffer[1];
unsigned long Adr = 0x00401004;

```

```

DWORD nCount = 1;

if ( LslGetDataExH(0, (void*)pBuffer, Adr, nCount ) == TRUE )
    printf( "LslGetDataEx( Addr=0x%08X ) = 0x%02X\n", Adr, pBuffer[0] );
else
    printf( "Error with LslGetDataEx\n" );

```

6.1.8.12 LslGetDataListEx

Corresponds to LslGetDataListEx() with 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

This function reads several data from the PLC. It tries to achieve the optimum speed. The SendDataList() / GetDataList() / GetData() functions are used for this purpose.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDataListExH(
    int32_t          ocbNum,
    const DWORD      dwCount,
    const DWORD      *pAddr,
    const DWORD      *pLen,
    unsigned char    *pResult
);

```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online connection block number (return value of OcbOpen())	
dwCount	const DWORD	Number of commands	
pAddr	const DWORD*	List of PLC addresses from where to read	
pLen	const DWORD*	List with the data lengths of the results	
pResult	CHAR*	List with the results	
Return parameters	Type	Description	
		TRUE	Fuction successful
		FALSE	error
		To obtain extended error information, call the GetLastError() function.	

Example

```
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])
{
    //Valid address in dwAddr[0]
    //Valid length in dwLen[0]
    dwLen[0] = 4;

    for(int i = 1; i < iMax; i++)
    {
        //Read every second element in the buffer "g_BigBuf"
        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("LslGetAddressVar failed(0x%08X)\n", GetLastError());
```

6.1.8.14 LslGetDataListSpecial

Corresponds to LslGetDataListSpecialH() with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDataListSpecial(
    const DWORD    dwCount,
    const DWORD    *pAddr,
    const DWORD    *pLen,
    char          *pResult
);
```

6.1.8.15 LslGetDataListSpecialH

This function executes several GetData commands in one.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDataListSpecialH(
    int32_t          ocbNum,
    const DWORD      dwCount,
    const DWORD*     *pAddr,
    const DWORD*     *pLen,
    char*            *pResult
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online connection block number (return value of OcbOpen())	
dwCount	const DWORD	Number of commands	
pAddr	const DWORD*	List of PLC addresses from where to read	
pLen	const DWORD*	List with the data lengths of the results	
pResult	CHAR*	List with the results	
Return parameters	Type	Description	
		TRUE	Fuction successful
		FALSE	error
		To obtain extended error information, call the GetLastError() function.	

Example

```
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])
{
    //Valid address in dwAddr[0]
    //Valid length in dwLen[0]
    dwLen[0] = 4;

    for(int i = 1; i < iMax; i++)
    {
        //Read every second element in the buffer "g_BigBuf"
        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("LslGetAdressVar failed(0x%08X)\n", GetLastError());

```

6.1.8.16 LslSetDataEx

Corresponds to LslSetDataExH() with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslSetDataEx(
    const uint8_t* pBuffer,
    DWORD          addr0,
    DWORD          nCount
);

```

6.1.8.17 LslSetDataExH

This function sends data to the PLC like SetData(), but without a length limit.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslSetDataExH(
    int32_t          ocbNum,
    const uint8_t*   pBuffer,
    DWORD            addr0,
    DWORD            nCount
);

```

Transfer parameters		Type	Description
ocbNum	int32_t	Online connection block number (return value of OcbOpen())	
pBuffer	const uint8_t*	Pointer to the data that should be sent.	
Addr0	DWORD	Address in the PLC	
nCount	DWORD	Number of bytes to send	
Return parameters		Type	Description
		TRUE Fuction successful	

		FALSE error
To obtain extended error information, call the GetLastError() function.		

Example

```
BOOL bResult = FALSE;
plcptr_t nObjAddr = 0;
uint32_t nCount = 1000000;
uint8_t buffer[1000000];

bResult = LslGetAdressVar("g_BigBuf", &nObjAddr);
if (bResult == TRUE && nObjAddr)
{
    //Valid address in nObjAddr
    //ToDo: Fill buffer

    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

Corresponds to [SendDataListH](#) function with ocbNum = 0

```
extern "C" LSL_BOOL LASAL32_EXPORTS SendDataList(
    uint16_t      length0,
    char          *data0,
    CB_FUNCTYPE   *callback
);
```

6.1.8.19 SendDataListH

The SendDataList() function sends a list of addresses to the PLC, whose content can be later collected through the [GetDownList\(\)](#) function and used.

```
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
);
```

Transfer parameters		Type	Description																
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())																	
Length0	uint16_t	Number of bytes in data0																	
data0	char*	Pointer to a buffer that contains the list of addresses. The list has the following format:																	
		<table border="1"> <thead> <tr> <th>Offset</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Unsigned short</td> <td>Number of address (max. 256)</td> </tr> <tr> <td>2</td> <td>Unsigned long</td> <td>1st address</td> </tr> <tr> <td>6</td> <td>Unsigned short</td> <td>Number of bytes in the 1st address</td> </tr> <tr> <td>8</td> <td>Unsigned long</td> <td>2nd address</td> </tr> <tr> <td>12</td> <td>Unsigned short</td> <td>Number of bytes in the 2nd address</td> </tr> </tbody> </table>		Offset	Type	Description	0	Unsigned short	Number of address (max. 256)	2	Unsigned long	1 st address	6	Unsigned short	Number of bytes in the 1 st address	8	Unsigned long	2 nd address	12
Offset	Type	Description																	
0	Unsigned short	Number of address (max. 256)																	
2	Unsigned long	1 st address																	
6	Unsigned short	Number of bytes in the 1 st address																	
8	Unsigned long	2 nd address																	
12	Unsigned short	Number of bytes in the 2 nd address																	
and so on																			
callback	CB_FUNCTYPE*	Pointer to a CB_FUNCTYPE callback function. When transferring the data to the PLC, it is possible to divide the data into blocks. The callback function is called before a block is transferred with the number of remaining blocks as a parameter. The callback function can be used, for example, to update a progress bar. If the callback parameter is NULL, no callback function is called.																	
Return parameters		Type	Description																
	LSL_BOOL	<table border="1"> <tr> <td>TRUE</td> <td>Success</td> </tr> <tr> <td>FALSE</td> <td>Error</td> </tr> </table>		TRUE	Success	FALSE	Error												
TRUE	Success																		
FALSE	Error																		
		To query extended error information, call GetLastError() .																	

Example

```

char DataList[256], *pB;
unsigned short len = 0;

pB = DataList;

// Number of addresses: 2
*(unsigned short *)pB = 2;
pB += sizeof(unsigned short);

// 1st.Address: 0x00401004

```

```

*(unsigned long *)pB = 0x00401004;
pB += sizeof(unsigned long);

// Number of bytes to the 1st address: 4
*(unsigned short *)pB = 4;
pB += sizeof(unsigned short);

// 2nd address: 0x00401008
*(unsigned long *)pB = 0x00401008;
pB += sizeof(unsigned long);

// Number of bytes to the 2nd address: 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

Corresponds to [SetDataH\(\)](#) function with ocbNum = 0

```

extern "C" LSL_BOOL LASAL32_EXPORTS SetData(
    const void    *pBuffer,
    uint32_t      addr0,
    uint16_t      nCount
);

```

6.1.8.21 SetDataH

The SetDataH() function sends data to the PLC.

```

extern "C" LSL_BOOL LASAL32_EXPORTS SetDataH(
    int32_t      ocbNum,
    const void   *pBuffer,
    uint32_t      addr0,
    uint16_t      nCount
);

```

Transfer parameters	Type	Description
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())
pBuffer	CONST VOID	Pointer to the data that should be sent
addr0	uint32_t	Address in the PLC
nCount	uint16_t	Number of bytes to set
Return parameters	Type	Description

		<p>TRUE Success</p> <p>FALSE Error</p> <p>To query extended error information, call the GetLastError() function.</p>
--	--	--------------------------------------------------------------------------------------------------------------------------------------

Example

```
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 Information and PLC Commands

6.1.9.1 GetBusType

Corresponds to GetBusTypeH() with ocbNum = 0.

```
extern "C" uint32_t LASAL32_EXPORTS GetBusType(void);
```

6.1.9.2 GetBusTypeH

Returns the currently set communication type.

```
extern "C" uint32_t LASAL32_EXPORTS GetBusTypeH(
    int32_t ocbNum
);
```

Transfer parameter	Type	Description	
ocbNum	int32_t	Online connection block number (return value of OcbOpen())	
Retrun parameter	Type	Description	
		Value	Name
		0	COMMUNICATION_NONE
			No communication

1	COMMUNICATION_V24	Communication via V24
2	COMMUNICATION_CAN	Communication via CAN bus
3	COMMUNICATION_TCP	Communication via TCP/IP
4	COMMUNICATION_MODEM	Communication via modem
5	COMMUNICATION_CAN_USB	Communication via the Lawicel CANUSB interface
6	COMMUNICATION_CAN_USB_FT	Communication via the Lawicel CANUSB interface (FTD2XX driver)
7	COMMUNICATION_VARAN32	Communication via VARAN32.DLL

Example

```
if (GetBusType() == COMMUNICATION_TCP)
{
    //Communication via TCP/IP
}
```

6.1.9.3 GetChk

Corresponds to GetChkH() with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetChk(
    void* pChk
);
```

6.1.9.4 GetChkByType

Corresponds to GetChkByTypeH() with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetChkByType(
    void* pChk,
    uint32_t modTypeMask,
    uint32_t modTypeEq
);
```

6.1.9.5 GetChkByTypeH

The function returns the checksum of the project. Only those modules that result in the value modTypeEq AND-linked with modTypeMask are taken into account in the calculation. The checksum is the sum of the module CRCs. The start value is 0, i.e. if no module is present, the checksum is 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetChkByTypeH(
    int32_t      ocbNum,
    void*        pChk,
    uint32_t     modTypeMask,
    uint32_t     modTypeEq
);
```

Transfer parameters		Type	Description	
ocbNum		int32_t	Online connection block number (return value of OcbOpen())	
pChk		void*	Pointer to a variable where the checksum of the project is entered (uint32_t)	
modTypeMask		uint32_t	Mask for the module type	
modTypeEq		uint32_t	Only this module type is taken into account	
Return parameters		Type	Description	
			TRUE	Fuction successful
			FALSE	Error
			To obtain extended error information, call the GetLastError() function.	

The following defines can be used to select the modules.

```
#define LSLMODTYPE_LOADER    0x00000001
#define LSLMODTYPE_APPL       0x00010000
```

Example 1 (calculate checksum for all modules)

```
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());
```

Example 2 (calculate checksum only for application modules)

```
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

The function returns the checksum of the project. Only those modules that result in the preset value of modTypeEq AND-linked with the presetting of modTypeMask are taken into account in the calculation. The pre-settings are set with the functions LsISetDfltModTypeParams() or LsISetDfltModTypeParamsH().

The checksum is the sum of the module CRCs. The start value is 0, i.e. if no module is present, the checksum is 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS GetChkH(
    int32_t    ocbNum,
    void*      pChk
);

```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online connection block number (return value of OcbOpen())	
pChk	void*	Pointer to a variable where the checksum of the project is entered (uint32_t)	
Return parameters	Type	Description	
		TRUE	Fuction successful
		FALSE	Error
		To obtain extended error information, call the GetLastError() function.	

Example

```

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

Corresponds to [GetCpuStatusH\(\)](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetCpuStatus(
    uint8_t *pStatus
);
```

6.1.9.8 GetCpuStatus2

Corresponds to GetCpuStatus2H() with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetCpuStatus2(
    uint8_t *pStatus,
    uint8_t direct
);
```

6.1.9.9 GetCpuStatus2H

The function queries the CPU status.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetCpuStatus2H(
    int32_t    ocbNum,
    uint8_t    *pStatus,
    uint8_t    direct
);
```

Transfer parameters		Type	Description	
ocbNum		int32_t	Online connection block number (return value of OcbOpen())	
pStatus		uint8_t*	Pointer to a byte variable that receives the CPU status codes of the PLC. A list of PLC status codes can be find in Appendix A .	
direct		uint8_t	0 A buffered (= outdated) status may be returned ≥0 The current status is obtained	
Return parameters		Type	Description	
			TRUE	Fuction successful
			FALSE	Error
			To obtain extended error information, call the GetLastError() function.	

Example

```
BOOL bOk = false;
```

```

uint8_t byCpuState = 39; // 39 = CS_OFFLINE
bool bCpuStateOk = FALSE;

// Retrieve current CPU status
bOk = GetCpuStatus2(&byCpuState, 1);
if (bOk)
{
    if (byCpuState == CS_RUN_RAM || byCpuState == CS_RUN_ROM)
    {
        bCpuStateOk = true;
    }
}

```

6.1.9.10 GetCpuStatusH

The GetCpuStatusH() function calls the CPU status codes from PLC.

```

extern "C" LSL_BOOL LASAL32_EXPORTS GetCpuStatusH(
    int32_t    ocbNum,
    uint8_t   *pStatus
);

```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())	
Pstatus	uint8_t	Pointer to the byte variable that contains the CPU status codes of the PLC. A list of CPU status codes can be find in appendix A.	
Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error
		To query extended error information, call the GetLastError() function.	

Example

```

BYTE Status;

if ( GetCpuStatus( &Status ) == TRUE )
    printf( "CpuStatus = 0x%02X\n", Status );
else
    printf( "Error in GetCpuStatus\n" );

```

6.1.9.11 GetPowerFailInfo

The function is obsolete.

```
extern "C" LSL_BOOL LASAL32_EXPORTS GetPowerFailInfo(
    uint8_t    *info,
    uint8_t    *errcode
);
```

Transfer parameters	Type	Description													
info...	uint8_t	Pointer to a variable where it is entered whether a PowerFail has occurred.	<table border="1"> <tr> <td>0</td><td>No PowerFail occurred or PowerFailInfo could not be determined (rc=FALSE)</td></tr> <tr> <td>1</td><td>PowerFail occurred</td></tr> </table>	0	No PowerFail occurred or PowerFailInfo could not be determined (rc=FALSE)	1	PowerFail occurred								
0	No PowerFail occurred or PowerFailInfo could not be determined (rc=FALSE)														
1	PowerFail occurred														
ErrCode	uint8_t	Pointer to a variable in which the cause of the error is entered if no PowerFail info could be determined (rc=FALSE).	<table border="1"> <tr> <td>0</td><td>No error</td></tr> <tr> <td>1</td><td>SIGMA driver could not be opened</td></tr> <tr> <td>2</td><td>Wrong version of the SIGMA driver</td></tr> <tr> <td>3</td><td>No SIGMATEK PC</td></tr> <tr> <td>4</td><td>UPS is not enabled</td></tr> <tr> <td>99</td><td>System error (out of memory, system-call failed, ...)</td></tr> </table>	0	No error	1	SIGMA driver could not be opened	2	Wrong version of the SIGMA driver	3	No SIGMATEK PC	4	UPS is not enabled	99	System error (out of memory, system-call failed, ...)
0	No error														
1	SIGMA driver could not be opened														
2	Wrong version of the SIGMA driver														
3	No SIGMATEK PC														
4	UPS is not enabled														
99	System error (out of memory, system-call failed, ...)														
Return parameters	Type	Description													
		TRUE	PowerFail info could be determined												
		FALSE	PowerFail info could not be determined												

Example

```
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

Returns a pointer to an ASCII 0 string. The string contains detailed information about the Lasal32.dll.

```
extern "C" const char* LASAL32_EXPORTS LslGetDllInfo(void);
```

Return Value

```

<DLLInfo>
  <DLL Name=Lasal32
    Version=01.01.149
    Build="dev"
    Platform="Win32"
    CompileDate="Jul 11 2023"
    CompileTime="11:18:57"
    DBG32LEVEL="0"
    DBGRLLLEVEL="0"
  </DLLInfo>

```

DBG32LEVEL: Debug level of the Lasal32.dll

DBGRLLLEVEL: Debug level of the refresh list

Example

```
const char *pInfo = LslGetDllInfo();
printf("DLLInfo: %s\n", pInfo);
```

6.1.9.13 LslGetDllVersion

Returns the DLL version number.

```
extern "C" uint32_t LASAL32_EXPORTS LslGetDllVersion(void);
```

Return Value

The version number has the following structure:

HIGH-WORD				LOW-WORD			
31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
				Major	Minor	Subversion	

Example

```
unsigned long DllVersion = LslGetDllVersion();
printf("DLL-Version: %d.%d.%d\n", DllVersion>>12 & 0x0000000F, DllVersion>>8 & 0x0000000F, DllVersion & 0x0000000F);
```

The DLL version 1.1.149 corresponds to the hex value 16#00001195.

6.1.9.14 LslGetMemInfo

Corresponds to LslGetMemInfoH() with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetMemInfo(
    uint8_t* pData,
    uint32_t len
);
```

6.1.9.15 LslGetMemInfoH

Provides start address, length, used size and attribute of memory areas of the PLC such as code area, RAM area, SRAM area. A length value < 0 is an error code.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetMemInfoH(
    int32_t ocbNum,
    uint8_t* pData,
    uint32_t len
);
```

Transfer parameters		Type	Description	
ocbNum		int32_t	Online connection block number (return value of OcbOpen())	
pData		uint8_t*	Pointer to the result buffer	
len		uint32_t	Length of the result buffer in bytes	
Return parameters		Type	Description	
			TRUE	Fuction successful
			FALSE	Error
			To obtain extended error information, call the GetLastError() function.	

The first four bytes of pData contain the number of elements of the subsequent array of structures (1 element per memory area). The data type of the structures is S_MEM_INFO. The attrib element specifies whether the start, size and used elements are valid or not.

```
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

```

Example

```

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;
            }
        }
    }
}

```

```

        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

Corresponds to [LslGetPLCInfoH](#) function with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetPLCInfo(
    uint8_t    *infostr,
    uint32_t   maxsize0
);

```

6.1.9.17 LslGetPLCInfoH

The `LslGetPLCInfoH()` function calls general information from the PLC, such as the version number, operating system, user data addresses and the user code area.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetPLCInfoH(
    int32_t      ocbNum,
    unsigned char *infostr,
    unsigned long maxsize0
);
```

Transfer parameters	Type	Description																																			
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())																																			
infostr		Pointer to a buffer that receives the PLC information. The PLC information has the following format: <table border="1" data-bbox="492 579 1036 1150"> <thead> <tr> <th>Offset</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Unsigned long</td> <td>User code start address</td> </tr> <tr> <td>4</td> <td>Unsigned long</td> <td>User code Lange</td> </tr> <tr> <td>8</td> <td>Unsigned long</td> <td>User data start address</td> </tr> <tr> <td>12</td> <td>Unsigned long</td> <td>User data length</td> </tr> <tr> <td>16</td> <td>Unsigned long</td> <td>(for internal use only)</td> </tr> <tr> <td>20</td> <td>Null-terminated string</td> <td>PLC-Name</td> </tr> <tr> <td>..</td> <td>Null-terminated string</td> <td>Graphic mode</td> </tr> <tr> <td>..</td> <td>Null-terminated string</td> <td>Version</td> </tr> <tr> <td>..</td> <td>Null-terminated string</td> <td>Boot image type ("OSBIN" = 386, "OSRTB" = IPCs)</td> </tr> <tr> <td>..</td> <td>Null-terminated string</td> <td>"LSE" if CPU has IPC graphic and is supported by the LSE kernel. 3 spaces when LSE is not supported.</td> </tr> </tbody> </table> Depending on the operating system version, additional information reserved for internal use remains undocumented.			Offset	Type	Description	0	Unsigned long	User code start address	4	Unsigned long	User code Lange	8	Unsigned long	User data start address	12	Unsigned long	User data length	16	Unsigned long	(for internal use only)	20	Null-terminated string	PLC-Name	..	Null-terminated string	Graphic mode	..	Null-terminated string	Version	..	Null-terminated string	Boot image type ("OSBIN" = 386, "OSRTB" = IPCs)	..	Null-terminated string	"LSE" if CPU has IPC graphic and is supported by the LSE kernel. 3 spaces when LSE is not supported.
Offset	Type	Description																																			
0	Unsigned long	User code start address																																			
4	Unsigned long	User code Lange																																			
8	Unsigned long	User data start address																																			
12	Unsigned long	User data length																																			
16	Unsigned long	(for internal use only)																																			
20	Null-terminated string	PLC-Name																																			
..	Null-terminated string	Graphic mode																																			
..	Null-terminated string	Version																																			
..	Null-terminated string	Boot image type ("OSBIN" = 386, "OSRTB" = IPCs)																																			
..	Null-terminated string	"LSE" if CPU has IPC graphic and is supported by the LSE kernel. 3 spaces when LSE is not supported.																																			
maxsize0		Defines the size of the PLC information buffer in bytes. The buffer should be sufficient in size (> 100 bytes).																																			
Return parameters	Type	Description																																			

		TRUE Success
		FALSE Error
To query extended error information, call the GetLastError() function.		

0-terminated String <Graphic mode>

```

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-terminated string <Version> must be interpreted as 3 hexadecimal values.

E.g. 1144 1.1.68

Example

```

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.PLCNAME      = %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: Bootimage Type    = %s\n", (char *)&strptr[2]);
strptr += (strlen(strptr) + 1);
if (strcmp((char *)strptr, "LSE") == 0)
    printf("LslGetPLCInfo: LSE Kernel supported");
else
    printf("LslGetPLCInfo: LSE Kernel not supported");
}

```

6.1.9.18 LslGetProjectInfo

Corresponds to [LslGetProjectInfoH](#) function with ocbNum = 0.

```
external "C" LSL_BOOL LASAL32_EXPORTS LslGetProjectInfo(void* pRetData0);
```

6.1.9.19 LslGetProjectInfoEx

Corresponds to [LslGetProjectInfoExH\(\)](#) with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetProjectInfoEx(
    PrjHeader* pRetData0,
    uint32_t    bufferSize,
    uint32_t*   nRead
);

```

6.1.9.20 LslGetProjectInfoExH

The function provides some information about the project. Only those modules that result in the preset value of modTypeEq AND-linked with the presetting of modTypeMask are taken into account. The pre-settings are set with the functions [LslSetDfltModTypeParams\(\)](#) or [LslSetDfltModTypeParamsH\(\)](#).

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetProjectInfoExH(
    int32_t      ocbNum,
    PrjHeader*   pRetData0,
    uint32_t     bufferSize,
    uint32_t*    nRead
);

```

Transfer parameters	Type	Description
ocbNum	int32_t	Online connection block number (return value of OcbOpen())
pRetData0	PrjHeader*	Pointer to the result buffer (see below for the structure of the result buffer)
BufferSize	uint32_t	Size of the result buffer pRetData0 in bytes. The buffer must be at least as large as a PrjHeader structure. However, it should be 8

		bytes larger so that a maximally long project name does not have to be shortened for error_master_count and ModTypes.
nRead	uint32_t*	Pointer to a variable where the number of data bytes in the result buffer is entered. This parameter may be NULL if the information is not required.
Return parameters	Type	Description
		<p>TRUE Function successful</p> <p>FALSE Error</p> <p>To obtain extended error information, call the GetLastError() function.</p>

Structure of the result buffer: first a PrjHeader structure. Followed by uint32_t error_master_count and, depending on the OS version, uint32_t ModTypes or 0 (see below).



error_master_count and ModTypes are copied directly after the project name and may therefore not be long-aligned.

Byte 0,1,2,3	Checksum of the project (application + loader)
Byte 4,5,6,7	Number of active modules
Byte 8	Number of pending messages
Byte 9,10,11	Dummy1, Dummy2, Dummy3
Byte 12-n	Project name (0-terminated)
Byte n+1,2,3,4	error_master_count (all (error) messages that have occurred)
Byte n+5,6,7,8	LasalOS version < 1.1.59: 0 LasalOS version ≥ 1.1.59: ModTypes (OR linking of the considered module types)

PrjHeader Structure

Offset	Type	Description
0	unsigned long	Checksum of the project (application + loader)
4	unsigned long	Number of active modules
8	char	MsgCounter

9	char	Dummy1
10	char	Dummy2
11	char	Dummy3
12	char [64]	Project name (0-terminated)

Example

```

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", 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

The LslGetProjectInfoH() function calls project information.

```

external "C" LSL_BOOL LASAL32_EXPORTS LslGetProjectInfoH(
    int32_t      ocbNum,
    PrjHeader   *pRetData0
);

```

Transfer parameters	Type	Description									
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen)									
pRetData0	PrjHeader	Pointer to the result buffer. Structure of result buffer see under LslGetProjectInfoExH() . <table border="1" data-bbox="490 1250 1039 1361"> <thead> <tr> <th>Offset</th><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Unsigned long</td><td>Project checksum</td></tr> <tr> <td>4</td><td>Unsigned long</td><td>Number of active module</td></tr> </tbody> </table>	Offset	Type	Description	0	Unsigned long	Project checksum	4	Unsigned long	Number of active module
Offset	Type	Description									
0	Unsigned long	Project checksum									
4	Unsigned long	Number of active module									

		8	Unsigned long	(for internal use only)
		12	Char [64]	Project name
		76	Char [180]	Reserved
Return parameters	Type	Description		
		TRUE	Success	
		FALSE	Error	
To query extended error information, call the GetLastError() function.				

Example

```

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

Corresponds to LslReadDateTimeH() with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslReadDateTime(
    LSL_SYSTEMTIME *pSysTime
);

```

6.1.9.23 Ls1ReadDateTimeH

The function returns the date and time of the PLC in pSysTime.

```
extern "C" LSL_BOOL LASAL32_EXPORTS Ls1ReadDateTimeH(
    int32_t      ocbNum,
    LSL_SYSTEMTIME *pSysTime
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online connection block number (return value of OcbOpen())	
pSysTime	LSL_SYSTEMTIME*	Pointer to an LSL_SYSTEMTIME structure where the values of the PLC are entered	
Return parameters	Type	Description	
		TRUE	Fuction successful
		FALSE	Error
		To obtain extended error information, call the GetLastError() function.	

Example

```
LSL_SYSTEMTIME SysTime;

if (Ls1ReadDateTime(&SysTime))
    printf("%d.%d.%d\n", SysTime.wDay, SysTime.wMonth, SysTime.wYear);
else
    printf("Ls1ReadDateTime() failed(0x%08X)\n", GetLastError());
```

6.1.9.24 Ls1SetDateTime

Corresponds to Ls1SetDateTimeH() with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS Ls1SetDateTime(
    LSL_SYSTEMTIME *pSysTime
);
```

6.1.9.25 Ls1SetDateTimeH

The function sets the date and time of the PLC to the values specified in pSysTime.

```
extern "C" LSL_BOOL LASAL32_EXPORTS Ls1SetDateTimeH(
    int32_t      ocbNum,
    LSL_SYSTEMTIME *pSysTime
);
```

Transfer parameters		Type	Description
ocbNum		int32_t	Online connection block number (return value of OcbOpen())
pSysTime		LSL_SYSTEMTIME*	Pointer to an LSL_SYSTEMTIME structure with date and time for the PLC
Return parameters		Type	Description
			<p>TRUE Function successful</p> <p>FALSE Error</p>
			To obtain extended error information, call the GetLastError() function.

Example

```
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

Corresponds to LslSyssernumCommandH() with 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

This function returns the serial number of the PLC or the serial number of the specified drive as a 0-terminated 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
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online connection block number (return value of OcbOpen())	
iCmd	int32_t	1	Read serial number of the PLC
		2	Read the serial number of the drive specified in plnput
plnput	void*	Pointer to the drive letter. The parameter has no meaning if iCmd is not 2.	
inLen	int32_t	Length of the data in plnput in bytes	
pOutput	void*	Pointer to a buffer where the serial number is entered as a 0-terminated ASCII string	
outLen	int32_t	Length of the buffer pOutput in bytes. The buffer must be at least 20 bytes long.	
Return parameters	Type	Description	
		TRUE	Fuction successful
		FALSE	Error
To obtain extended error information, call the GetLastError() function.			

Example

```

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 [SetCommandH](#)-Funktion mit ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS SetCommand(
    uint8_t uCommand
);
```

6.1.9.29 SetCommandH

The SetCommand() function sends a command to the PLC.

```
extern "C" LSL_BOOL LASAL32_EXPORTS SetCommandH(
    int32_t ocbNum,
    uint8_t uCommand
);
```

Transfer parameters		Type	Description				
ocbNum		int32_t	Online Connection Block number (return value of OcbOpen())				
uCommand		uint8_t	<p>This parameter indicates the command type. Only the following commands are valid:</p> <ul style="list-style-type: none"> • Reset (3) • Run (4) 				
Return parameters		Type	Description				
		LSL_BOOL	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; padding: 2px;">TRUE</td><td style="width: 50%; padding: 2px;">Success</td></tr> <tr> <td style="padding: 2px;">FALSE</td><td style="padding: 2px;">Error</td></tr> </table> <p>To query extended error information, call the GetLastError() function.</p>	TRUE	Success	FALSE	Error
TRUE	Success						
FALSE	Error						

Example

```
if ( SetCommandH(0, 3) == TRUE )
    printf( "SetCommand( 3 ) O.K. (RESET)\n" );
else
    printf( "Error in SetCommand( 3 ) (RESET)\n" );
```

6.1.10 Administrative Functions

6.1.10.1 SetMultiThreadSupport

The SetMultiThreadSupport() function can be used to enable or disable multi-thread support for the online functions. With multi-thread support enabled, all online functions are protected with a critical section object.

In single-thread programming, multi-thread support is disabled. The effort of protecting the online functions is thereby reduced. Please note that the online functions are not thread-safe when multi-thread support is disabled.

Multi-thread support is enabled by default.

```
extern "C" void WINAPI SetMultiThreadSupport(unsigned long val);
```

Transfer parameters	Type	Description				
val		<p>Indicates whether the multi-thread support is enabled</p> <table border="1"> <tr> <td>0</td><td>Disabled</td></tr> <tr> <td>1</td><td>Enabled</td></tr> </table>	0	Disabled	1	Enabled
0	Disabled					
1	Enabled					

It should be noted that in Lasal32.dll versions before 1.34, the application is responsible for protecting the functions from a new entry in multi-thread applications.

Requirements

Version Lasal32.dll version 1.34 or higher is required.

6.1.11 File Functions

6.1.11.1 CB_FORMAT_FUNCTYPE

Callback function of [LslFormatDrive\(\)](#).

```
typedef long __cdecl CB_FORMAT_FUNCTYPE(
    LSL32_FORMAT_PROGRESS_INFO * pFormatProgressInfo);
```

Transfer parameters	Type	Description
PFormatProgressInfo		Pointer to an LSL32_FORMAT_PROGRESS_INFO type that contains the progress information
Return parameters	Type	Description
		As long as the return value is greater than 0, the DLL calls the callback function

Information Structure

```

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	Name of device
Total	DWORD	Number of sectors that must be processed
Completed	DWORD	Number of sectors that have been successfully processed
Status	DWORD	Status information on the formatting process. <ul style="list-style-type: none"> • LSL32_FORMAT_PROGRESS_READY The drive is ready to format (used internally). • LSL32_FORMAT_PROGRESS_BUSY The formatting is running. • LSL32_FORMAT_PROGRESS_DONE Formatting completed successfully. • LSL32_FORMAT_PROGRESS_ERROR Formatting error

Requirements

Lasal32.dll: 01.01.018 and higher

LasalOS: 01.01.067 and higher

Example

```

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

Corresponds to [FileInfoH\(\)](#) function with ocbNum = 0.

```

LSL_BOOL LASAL32_EXPORTS FileInfo(
    char          *Rdbuf,
    const char    *dest,
    uint32_t      buflen
);

```

6.1.11.3 FileInfoH

The FileInfoH() function returns information on the file in the PLC.

```

extern "C" LSL_BOOL LASAL32_EXPORTS FileInfo(
    int32_t      ocbNum,
    char          *Rdbuf,
    char          *dest,
    unsigned long buflen
);

```

Transfer parameters	Type	Description
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())
Rdbuf	CHAR	Pointer to a buffer that contains the information. This pointer must have the following structure: <code>typedef struct LSL_FILE_INFO</code> <code>{</code>

		<pre> char FileName[8]; char Extension[3]; uint8_t cAttributes; _DDE_DATIM dtDateTime; // of last modification uint32_t lFileSize; } LSL_FILE_INFO; </pre> <p>The dtDateTime element is filled with the following information:</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	Pointer to a 0-terminated string that contains the name of the file in the PLC. An absolute path name should always be specified, which consists of the drive, directory and the file name.						
buflen	UNSIGNED LONG	Returns the buffer length						
Return parameters	Type	Description						
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td><td style="width: 10%; text-align: center;">TRUE</td><td>Success</td></tr> <tr> <td></td><td style="text-align: center;">FALSE</td><td>Error</td></tr> </table>		TRUE	Success		FALSE	Error
	TRUE	Success						
	FALSE	Error						
		To query extended error information, call the GetLastError() function.						

Example

```

LSL_FILE_INFO fileInfo;
if (FileInfo((char*)fileInfo, "C:\\\\testdir", sizeof(fileInfo)))
{
    //File name and extension
    printf("Attributes of %.8s.%s: ", 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

Corresponds to [FileLoadH\(\)](#) function with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS FileLoad(
    char          *RXbuf,
    const char    *dest,
    uint32_t       datalen,
    CB_FUNCTYPE   *callback,
    int32_t        offs
);

```

6.1.11.5 FileLoadH

The FileLoadH() function reads data from a file on the PLC starting from the position defined by the offs parameter.

```

typedef unsigned long _cdecl CB_FUNCTYPE(unsigned long);

extern "C" LSL_BOOL LASAL32_EXPORTS FileLoadH(
    int32_t        ocbNum,
    char          *RXbuf,
    char          *dest,
    unsigned long  datalen,
    CB_FUNCTYPE   *callback,
    long           offs
);

```

Transfer parameters	Type	Description
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())
RXbuf	CHAR	Pointer to the buffer that transfers the data from the file to the PLC
dest	CHAR	Pointer to a 0-terminated string that contains the name of the file in the PLC. An absolute path name should always be specified, which consists of the drive, directory and the file name.
files	UNSIGNED LONG	Indicates the number of bytes to read from the file. If the given datalen exceeds the length of the file to be read, the function is not executed.
callback	CB_FUNCTYPE*	Pointer to a CB_FUNCTYPE Callback function. With the data transfer to the PLC, it is possible that the data is divided into blocks. The callback function is called before the application of the block with the number of blocks remaining as the input parameter. The

		callback function can be used, for example, to update a progress bar. If the callback parameter is NULL, no callback function is called.
offs	LONG	Provides the 0 position in the file, from which to start reading the data
Return parameters	Type	Description
		<div style="display: flex; justify-content: space-between; align-items: center;"> <div style="flex: 1; padding-right: 10px;"> <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="flex: 1; padding-right: 10px;">TRUE</div> <div>Success</div> </div> </div> <div style="flex: 1; padding-right: 10px;"> <div style="display: flex; justify-content: space-between; align-items: center;"> <div style="flex: 1; padding-right: 10px;">FALSE</div> <div>Error</div> </div> </div> <div style="flex: 1; padding-right: 10px;"> <p>To query extended error information, call the GetLastError() function.</p> </div> </div>

[FileInfo](#) or [FileInfoH\(\)](#) can be used to query the length of the file in the PLC.

Example

```

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: processing data
        }
    }
}

```

```
    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

Corresponds to the [FileSaveH\(\)](#) function with 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

Corresponds to [FileSaveExH\(\)](#) function with 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

The FileSaveExH() function loads data in a file into the PLC. If the specified file already exists and the attribute read-only is not set, the file will be overwritten. This function returns an error code in the event of a remote error.

```
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	Description															
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())															
dest	CONST CHAR	Pointer to a 0-terminated string that contains the name of the file in the PLC. An absolute path name should always be specified, which consists of the drive, directory and the file name.															
pData	CONST CHAR	Pointer to the buffer with the data that should be written to the file															
len	uint32_t	Defines the number of bytes that should be written to the file															
callback	CB_FUNCTYPE*	Pointer to a CB_FUNCTYPE callback function. With the data transfer to the PLC, it is possible that the data is divided into blocks. The callback function is called before the transfer of a block with the number of blocks remaining as the input parameter. The callback function can be used, for example, to update a progress bar. If the callback parameter is NULL, no callback function is called.															
errCodeRemote	int32_t	<p>Pointer to a variable that contains the error code of the remote PLC. This parameter is only then practical, when the function is defective and GetLastError() returns ERROR_SIGMA32_REMOTE_ERROR.</p> <p>It should be noted that operation system versions older than 5.53 do not register an error code. In this case, the FileSave function set the value of errCodeRemote to ERROR_SIGMA32_REMOTE_ERROR. This is an unspecified error code. For a list of the possible LasalOS error codes, see the LasalOS documentation.</p> <p>The following table contains a list of the LasalOS error codes. It is important to note that these error codes are only an excerpt of all possible error codes. For a list of the other possible error codes, see the LasalOS documentation.</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_GENE RAL</td></tr> <tr> <td>0xA00100 00</td><td>General file error, not categorized.</td><td>LSLOS_FILE_ERROR_G ENERAL</td></tr> <tr> <td>0xA00180 03</td><td>The file name, directory name or volume label syntax is incorrect.</td><td>LSLOS_ERROR_INVALI D_FILENAME</td></tr> <tr> <td>0xA00180 04</td><td>The system cannot find the specified drive.</td><td>LSLOS_ERROR_DRIVE _NOT_FOUND</td></tr> </tbody> </table>	Code	Description	Name	0xA00080 13	General error, not categorized.	LSLOS_ERROR_GENE RAL	0xA00100 00	General file error, not categorized.	LSLOS_FILE_ERROR_G ENERAL	0xA00180 03	The file name, directory name or volume label syntax is incorrect.	LSLOS_ERROR_INVALI D_FILENAME	0xA00180 04	The system cannot find the specified drive.	LSLOS_ERROR_DRIVE _NOT_FOUND
Code	Description	Name															
0xA00080 13	General error, not categorized.	LSLOS_ERROR_GENE RAL															
0xA00100 00	General file error, not categorized.	LSLOS_FILE_ERROR_G ENERAL															
0xA00180 03	The file name, directory name or volume label syntax is incorrect.	LSLOS_ERROR_INVALI D_FILENAME															
0xA00180 04	The system cannot find the specified drive.	LSLOS_ERROR_DRIVE _NOT_FOUND															

		0xA00180 09	The system cannot find the specified file.	LSLOS_ERROR_FILE_NOT_FOUND
		0xA00180 10	Access was denied.	LSLOS_ERROR_ACCESS_DENIED
		0xA00180 13	The system cannot find the specified path.	LSLOS_ERROR_PATH_NOT_FOUND
		0xA00180 16	Insufficient space on the hard drive.	LSLOS_ERROR_DISK_FULL
		0xA00180 1A	The media on the drive has possibly changed.	LSLOS_ERROR_MEDIA_CHANGED
		0xA00180 1D	The module is not ready.	LSLOS_ERROR_NOT_READY
		0xA00180 1E	The data carrier is write protected.	LSLOS_ERROR_WRITE_PROTECT
Return parameters	Type	Description		
		TRUE	Success	
		FALSE	Error	
		To query extended error information, call the GetLastError() function.		

Requirements

Version Lasal32.dll 1.36 or higher required.

Example

```
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

The FileSaveH() function loads data in a file into the PLC. If the specified file is available, its contents are destroyed.

In the event of a remote error, this function does not provide an error code from the PLC. Use the [FileSaveExH\(\)](#) function if PLC error codes are desired.

```
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
);

```

Transfer parameters	Type	Description									
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())									
dest	CONST CHAR	Pointer to a null-terminated string that contains the name of the file in the PLC. An absolute path name should always be specified, which consists of the drive, directory and the file name.									
pData	CHAR	Pointer to the buffer with the data that should be written to the file									
len	uint32_t	Defines the number of bytes that should be written to the file									
callback	CB_FUNCTYPE*	Pointer to a CB_FUNCTYPE callback function. With the data transfer to the PLC, it is possible that the data is divided into blocks. The callback function is called before the transfer of a block with the number of blocks remaining as the input parameter. The callback function can be used, for example, to update a progress bar. If the callback parameter is NULL, no callback function is called.									
attrib	uint32_t	Defines the attributes for the file. A combination of the following attributes are possible:									
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 30%;">Attribute</th> <th style="width: 70%;">Meaning</th> </tr> </thead> <tbody> <tr> <td>LSL32_ATTR_READONLY (0x01)</td> <td>The file can only be read.</td> </tr> <tr> <td>LSL32_ATTR_HIDDEN (0x02)</td> <td>The file is hidden. It is not visible in a normal directory.</td> </tr> <tr> <td>LSL32_ATTR_SYSTEM (0x04)</td> <td>The file is a part of the operating system or is used by the operating system exclusively.</td> </tr> <tr> <td>LSL32_ATTR_ARCHIVE (0x20)</td> <td>The file should be archived. Applications use this attribute to mark files for storage or backups.</td> </tr> </tbody> </table>		Attribute	Meaning	LSL32_ATTR_READONLY (0x01)	The file can only be read.	LSL32_ATTR_HIDDEN (0x02)	The file is hidden. It is not visible in a normal directory.	LSL32_ATTR_SYSTEM (0x04)	The file is a part of the operating system or is used by the operating system exclusively.	LSL32_ATTR_ARCHIVE (0x20)	The file should be archived. Applications use this attribute to mark files for storage or backups.
Attribute	Meaning										
LSL32_ATTR_READONLY (0x01)	The file can only be read.										
LSL32_ATTR_HIDDEN (0x02)	The file is hidden. It is not visible in a normal directory.										
LSL32_ATTR_SYSTEM (0x04)	The file is a part of the operating system or is used by the operating system exclusively.										
LSL32_ATTR_ARCHIVE (0x20)	The file should be archived. Applications use this attribute to mark files for storage or backups.										
Starting from version 5.28, this parameter is ignored!											
Return parameters	Type	Description									
		TRUE	Success								
		FALSE	Error								

		To query extended error information, call the GetLastError() function.
--	--	----------------------------------------------------------------------------------------

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

Corresponds to FileSetAttributesH() with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS FileSetAttributes(
    const char* dest,
    uint32_t     attrib
);

```

6.1.11.11 FileSetAttributesH

This function sets the file attributes. Attributes that are not specified in attrib are deleted. FileInfoH() returns the file attributes.

```

extern "C" LSL_BOOL LASAL32_EXPORTS FileSetAttributesH(
    int32_t     ocbNum,
    const char* dest,
    uint32_t     attrib
);

```

Transfer parameters	Type	Description				
ocbNum	int32_t	Online connection block number (return value of OcbOpen())				
dest:	const char*	Pointer to a 0-terminated string that specifies the name of the file in the PLC. An absolute path name should always be specified, consisting of the drive, the directory and the file name.				
attrib;	uint32_t	<p>Shows the file attributes for the file. The RTK operating system supports a combination of the following file attributes:</p> <ul style="list-style-type: none"> • LSL32_ATTR_READ_ONLY • LSL32_ATTR_HIDDEN • LSL32_ATTR_SYSTEM • LSL32_ATTR_ARCHIVE <p>All other operating systems only support the file attribute LSL32_ATTR_READ_ONLY.</p> <p>The file attribute LSL32_ATTR_DIR can only be read for all operating systems.</p>				
Return parameters	Type	Description				
		<table border="1"> <tr> <td>TRUE</td><td>Fuction successful</td></tr> <tr> <td>FALSE</td><td>Error</td></tr> </table>	TRUE	Fuction successful	FALSE	Error
TRUE	Fuction successful					
FALSE	Error					

		To obtain extended error information, call the GetLastError() function.
--	--	-----------------------------------------------------------------------------------------

Example

```
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 and Salamander and others
    fileInfo.cAttributes |= LSL32_ATTR_HIDDEN; //only RTK
    fileInfo.cAttributes |= LSL32_ATTR_SYSTEM; //only RTK
    fileInfo.cAttributes |= LSL32_ATTR_ARCHIVE; //only 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

Corresponds to [LsICheckDiskH\(\)](#) function with 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
);
```

Example

```
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

Function for testing an FAT formatted logic drive. Optionally, file system errors can be corrected automatically.

```

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
);

```

Transfer parameters	Type	Description
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())
DriveName	CONST CHAR	Pointer to the drive letter
ulOption	uint32_t	Optional function 0 =>No correction when errors are found in the file system. #define LSL32_CHKDSK_OPTION_CORRECT 0x01 => CheckDisk corrects file system errors.
ulFlags	uint32_t	Reserved
CheckDiskProgressInfo	CB_CHKDSK_FUNC_TYPE	Pointer to a CB_CHKDSK_FUNCTYPE callback function used for displaying progress information. If no information is required, this function can be null.
pErrors	uint32_t	Pointer to a variable that contains the number of detected errors
pResult	int32_t	Pointer to an INT value that is filled with the operation result. ≥0 Success <0 Negative error code
Return parameters	Type	Description
		TRUE Success FALSE Error
		To query extended error information, call the GetLastError() function.

Callback Function

With errors, CheckDisk is called recursively until all errors have been corrected; since when an error is corrected, it can lead to new errors. This is run through the DLL and can be monitored in the callback function.

```
typedef long __cdecl CB_CHKDSK_FUNCTYPE
    (char * strProgressInfo, unsigned long ProgressInfoLen);
```

Transfer parameters		Type	Description
strProgressInfo	CHAR	Pointer to a string that contains the information	
ProgressInfoLen	UNSIGNED LONG	Information length	
Return parameters		Type	Description
		≥0 Callback function is called from the DLL <0 Callback function is no longer called	

Requirements

Lasal32.dll: 01.01.019

LasalOS: 01.01.079

6.1.11.14 LslFileDelete

Corresponds to [LslFileDeleteH\(\)](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileDelete(
    const char    *ccFileDirectoryName,
    uint32_t      ulOption,
    int32_t       *pResult
);
```

6.1.11.15 LslFileDeleteH

LslFileDeleteH deletes a specific file or removes a specific directory.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileDeleteH(
    int32_t       ocbNum,
    const char    *ccFileDirectoryName,
    uint32_t      ulOption,
    int32_t       *pResult
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())	
ccFileName	CONST CHAR	Pointer to a file or directory	
ulOption	uint32_t	Delete file or remove directory. <code>#define LSL32_DELETE_FILE 0</code> <code>#define LSL32_REMOVE_DIR 1</code>	
pResult	int32_t	Pointer to an INT value that is filled with the operation result <div style="display: flex; justify-content: space-between; align-items: center;"> ≥0 Success </div> <div style="display: flex; justify-content: space-between; align-items: center;"> <0 Negative error code </div>	
Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error
		To query extended error information, call the GetLastError() function.	

Requirements

Lasal32.dll	01.01.018 and higher
LasalOS	01.01.067 and higher

Example

```

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

LslFindClose closes the previous calls to [LslFindFirst\(\)](#)/[LslFindNext\(\)](#).

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFindCloseH(
    int32_t    ocbNum,
    int32_t    *pResult
);
```

Transfer parameters		Type	Description				
ocbNum		int32_t	Online Connection Block number (return value of OcbOpen())				
pResult		int32_t	Pointer to an INT value that is filled with the operation result <table border="1" data-bbox="470 491 1019 595"> <tr> <td>≥0</td><td>Success</td></tr> <tr> <td><0</td><td>Negative error code</td></tr> </table>	≥0	Success	<0	Negative error code
≥0	Success						
<0	Negative error code						
Return parameters		Type	Description				
			<table border="1" data-bbox="504 650 1019 737"> <tr> <td>TRUE</td><td>Success</td></tr> <tr> <td>FALSE</td><td>Error</td></tr> </table> To query extended error information, call the GetLastError() function.	TRUE	Success	FALSE	Error
TRUE	Success						
FALSE	Error						

6.1.11.17 LslFindFirstH

LslFindFirstH() searches a directory for a file that meets specific criteria.

```
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
);
```

Transfer parameters		Type	Description
ocbNum		int32_t	Online Connection Block number (return value of OcbOpen())
namePattern		CONST CHAR	Pointer to a file name that can contain wildcard characters "*" (match with null or multiple characters) or "?" (compare exactly this character) and can optionally predefined a path. If a path already exists, it cannot contain any wildcard characters. For compatibility

		with the MS DOS pattern, "*.*" is converted to "*". To search for file names that fill one or more points, use "?*.*" or "*.*?*".				
attr	BYTE	Outputs a set of all file attributes, which needs a file to compare the query. Each combination of the following flags can be defined for the Attr and AttrMask parameters: <ul style="list-style-type: none"> • LSL32_ATTR_READ_ONLY • LSL32_ATTR_HIDDEN • LSL32_ATTR_SYSTEM • LSL32_ATTR_DIR • LSL32_ATTR_ARCHIVE 				
attrMask	BYTE	Defines the set of attributes, which assigned with Attr. Attributes defined in Attr are added in AttrMask automatically. Specify attributes that should not be used in AttrMask. The pseudo attribute RTF_FIND_NO_ALIAS has no effect on AttrMask.				
pResult	INT	Pointer to an int value that is filled with the operation result. When successful, this value is greater than or equal to 0. If the process fails, a negative error code is returned.				
fileInfo	_DDE_INFO	Pointer to a _DDE_INFO structure. If the function is successful and pResult also shows a valid result, this structure is filled with the directory of located files. This parameter can be NULL.				
fileName	CHAR	Pointer to a string buffer, which contains the file names without the path when a file is located.				
maxLength	UINT	Size of the buffer displayed by FileName in bytes. File name with a length greater than maxLength -1 are not found.				
Return parameters	Type	Description				
		<table border="1"> <tr> <td>TRUE</td><td>Success, command can be loaded into the target system and run</td></tr> <tr> <td>FALSE</td><td>Error</td></tr> </table> <p>To query extended error information, call the GetLastError() function.</p>	TRUE	Success, command can be loaded into the target system and run	FALSE	Error
TRUE	Success, command can be loaded into the target system and run					
FALSE	Error					



It is important to close `LsIFindFirstH()`/`LsIFindNextH()` calls with `LsIClose` when information is no longer required. `LsIFindFirst()` closes a previous `LsIFindFirst()` call. The operating system supports one open `LsIFindFirstH()` only.

6.1.11.18 LslFindFirst, LslFindNext, LslFindClose

Corresponds to the [LslFindFirstH\(\)](#), [LslFindNextH\(\)](#), [LslFindCloseH\(\)](#) function with the 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

LslFindNextH finds other files with the same search criteria as the previous call to [LslFindFirst\(\)](#).

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFindNextH(
    int32_t          ocbNum,
    int              *pResult,
    _DDE_INFO        *fileInfo,
    char              *fileName,
    UINT             maxLength
);
```

Transfer parameters	Type	Description
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())
pResult	int	Pointer to an INT value that is filled with the operation result <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> ≥ 0 Success < 0 Negative error code </div>
fileInfo	_DDE_INFO	Pointer to a _DDE_INFO structure. If the function is successful and pResult also shows a valid result, this structure is filled with the directory of located files. This parameter can be NULL.

fileName	char	Pointer to a string buffer, which contains the file names without the path when a file is located				
maxLength	UINT	Size of buffer displayed by FileName in bytes. File name with a length greater than maxLength -1 are not found.				
Return parameters	Type	Description				
		<table> <tr> <td>TRUE</td><td>Success</td></tr> <tr> <td>FALSE</td><td>Error</td></tr> </table> <p>To query extended error information, call the GetLastError() function.</p>	TRUE	Success	FALSE	Error
TRUE	Success					
FALSE	Error					



It is important to close the active [LslFindFirstH\(\)](#)/[LslFindNextH](#) functions when information is no longer required.

6.1.11.20 LslFileTransfer

Corresponds to [LslFileTransferH\(\)](#) function with 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

Corresponds to [LslFileTransferFlushH\(\)](#) with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferFlush(void);
```

6.1.11.22 LslFileTransferFlushH

Triggers a cache flush of the file system on the PLC. Use this function after file downloads with the [LslFileTransferH\(\)](#) function if the **LSL_FILETRANSFER_DONT_USE_AUTO_FLUSH** option is set.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferFlushH(
    int32_t    ocbNum,
);
```

Transfer parameters		Type	Description
ocbNum		int32_t	Online connection block number (return value of OcbOpen())
Return parameters		Type	Description
			<p>TRUE Function successful</p> <p>FALSE Error</p> <p>With LslGetError, the last error code can be determined. This code corresponds either to a Lasal32 or system specific error code (Windows: GetLastError or Linux: errno).</p>

6.1.11.23 LslFileTransferH

The LslFileTransferH() function transfers files between the PC and the control. Unlike the FileSave() function, the LslFileTransferH() function is error-tolerant with low quality connections. For new projects the FileTransfer() function should be used only. The function returns a PLC error code in the event of a remote error.

```
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
);
```

Transfer parameters		Type	Description
ocbNum		int32_t	Online Connection Block number (return value of OcbOpen())
direction		uint32_t	Indicates the direction in which the file should be transferred (from or to the control)
		Attribute	Meaning
		LSL_FILETRANSFER_DIRECTION_TO_PLA	The file is transferred from the PC to the control.

		LSL_FILETRANSFE R_DIRECTION_FR OMPLC	The file is transferred from the control to the PC.										
fileSrc	const char	Pointer to the name of the source file											
fileDest	const char	Pointer to the name of the destination file											
flags	uint32_t	Flags used to open the file <table border="1" style="margin-top: 10px;"> <thead> <tr> <th>Attribute</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>LSL_OPENFLAGS_RDONLY</td> <td>Opens the file for read only</td> </tr> <tr> <td>LSL_OPENFLAGS_WRONLY</td> <td>Opens file for writing only</td> </tr> <tr> <td>LSL_OPENFLAGS_CREATE</td> <td>Opened file is readable and writable. The file is created if it does not already exist.</td> </tr> <tr> <td>LSL_FILETRANSFE R_DONT_USE_AUT O_FLUSH</td> <td> Option to quickly download many files or directories with LsFileTransferH() from the PC to the PLC by preventing automatic file system cache flushes. After all files have been downloaded, it is recommended to call LsFileTransferFlushH(). If the operating system does not support this option, LsFileTransferH() does NOT fail, but performs automatic cache flushes. </td> </tr> </tbody> </table>		Attribute	Meaning	LSL_OPENFLAGS_RDONLY	Opens the file for read only	LSL_OPENFLAGS_WRONLY	Opens file for writing only	LSL_OPENFLAGS_CREATE	Opened file is readable and writable. The file is created if it does not already exist.	LSL_FILETRANSFE R_DONT_USE_AUT O_FLUSH	Option to quickly download many files or directories with LsFileTransferH() from the PC to the PLC by preventing automatic file system cache flushes. After all files have been downloaded, it is recommended to call LsFileTransferFlushH(). If the operating system does not support this option, LsFileTransferH() does NOT fail, but performs automatic cache flushes.
Attribute	Meaning												
LSL_OPENFLAGS_RDONLY	Opens the file for read only												
LSL_OPENFLAGS_WRONLY	Opens file for writing only												
LSL_OPENFLAGS_CREATE	Opened file is readable and writable. The file is created if it does not already exist.												
LSL_FILETRANSFE R_DONT_USE_AUT O_FLUSH	Option to quickly download many files or directories with LsFileTransferH() from the PC to the PLC by preventing automatic file system cache flushes. After all files have been downloaded, it is recommended to call LsFileTransferFlushH(). If the operating system does not support this option, LsFileTransferH() does NOT fail, but performs automatic cache flushes.												
callback	CB_FUNCTYPE2*	Pointer to a CB_FUNCTYPE2 callback function. When transferring the file, it is possible to divide the data into blocks. Before the file is transferred, the callback function is called once with the file length in bytes and once with the number of remaining blocks as parameters. The callback function is called during the transfer of the file with the number of bytes that have already been copied as a parameter. The callback function can be used, for example, to update a progress bar. If the callback parameter is NULL, no callback function is called.											
pUser	void	User pointer sent to the callback function (if used)											
Return parameters	Type	Description											
		TRUE	Success										
		FALSE	Error										

		To query extended error information, call the GetLastError() function.
--	--	----------------------------------------------------------------------------------------

Requirements

Version: Lasal32.dll 01.01.109 or higher

Example

```
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

Corresponds to [LslFileTransferFromPlcBufH](#) function with 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

The LslFileTransferFromPlcBufH function can be used to copy the contents of a file from the control directly into a buffer.

```

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
);

```

Transfer parameters	Type	Description
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())
FileSrc	CONST CHAR	Pointer to the file name in the control
DestBuf	uint8_t	Pointer to the destination buffer for the incoming data
Size	uint32_t	Size of the destination buffer
Getsize	uint32_t*	Optional pointer to a variable where the total number of transferred bytes is stored
callback	CB_FUNCTYPE2*	Pointer to a CB_FUNCTYPE2 callback function. When transferring the file, it is possible to divide the data into blocks. Before the file is transferred, the callback function is called once with the file length in bytes and once with the number of remaining blocks as parameters. The callback function is called during the transfer of

		the file with the number of bytes that have already been copied as a parameter. The callback function can be used, for example, to update a progress bar. If the callback parameter is NULL, no callback function is called.				
pUser	void*	Optional user pointer sent to the callback function				
Return parameters	Type	Description				
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; padding: 2px;">TRUE</td><td style="width: 50%; padding: 2px;">Success</td></tr> <tr> <td style="padding: 2px;">FALSE</td><td style="padding: 2px;">Error</td></tr> </table> <p>To query extended error information, call the GetLastError() function.</p>	TRUE	Success	FALSE	Error
TRUE	Success					
FALSE	Error					

Requirements

Version: Lasal32.dll 01.01.109 or higher

Example

```

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("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

Corresponds to LslFileTransferMakeDirH() with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferMakeDir(
    const char*  dirName,
    uint32_t     flags
);

```

6.1.11.27 LslFileTransferMakeDirH

This function creates a directory with the specified path. The command will fail if it is not supported by the operating system.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslFileTransferMakeDirH(
    int32_t      ocbNum,
    const char*  dirName,
    uint32_t     flags
);

```

Transfer parameters		Type	Description
ocbNum		int32_t	Online connection block number (return value of OcbOpen())
dirname		const char*	Pointer to a 0-terminated string that specifies the complete path, e.g: C:\temp\test1\result
flags		uint32_t	Bit 0 (recursive): Create directories recursively (1) or not (0) Bit 1 (dontSync): All non-empty buffers in the kernel are written to the device (0) or not (1)
Return parameters		Type	Description
			<p>TRUE Fuction successful</p> <p>FALSE Error</p> <p>To obtain extended error information, call the GetLastError() function.</p>

Example

```
//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

Corresponds to the [LslFileTransferToPlcBufH](#) function with 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

The FileTransferToPlcBufH() function can be used to transfer data to the control. This data are written directly in a defined file. There is no reason to create a local file.

```
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
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())	
SrcData	CONST VOID	Pointer to the data to transfer	
Size	uint32_t	Data length	
FileDest	CONST CHAR	Pointer to the file name in the control	
Flags	uint32_t	File flags	
Callback	CB_FUNCTYPE2*	<p>Pointer to a CB_FUNCTYPE2 callback function. When transferring the file, it is possible to divide the data into blocks. Before the file is transferred, the callback function is called once with the file length in bytes and once with the number of remaining blocks as parameters. The callback function is called during the transfer of the file with the number of bytes that have already been copied as a parameter.</p> <p>The callback function can be used, for example, to update a progress bar. If the callback parameter is NULL, no callback function is called.</p>	
PUser	VOID	Optional user pointer sent in the callback function	
Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error
		To query extended error information, call the GetLastError() function.	

Requirements

Version: Lasal32.dll 01.01.109 or higher

Example

See [LslFileTransferFromPlcBufH\(\)](#).

6.1.11.30 LslFormatDrive

Corresponds to [LslFormatDriveH\(\)](#) function with 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

The LslFormatDriveH() function formats the specified drive.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslFormatDriveH(
    int32_t          ocbNum,
    unsigned char    ucDriveLetter,
    unsigned long    ulOption,
    int              *pResult,
    CB_FORMAT_FUNCTYPE FormatInfoCallback
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())	
ucDriveLetter	UNSIGNED CHAR	Drive letter of the hard drive to format	
ulOption	UNSIGNED LONG	Reserved	
pResult	INT	Pointer to an INT value that is filled with the operation result	
		≥ 0 Success < 0 Negative error code	
FormatInfoCallback	CB_FORMAT_FUNC_TYPE	Pointer to a CB_FORMAT_FUNCTYPE Callback function that shows information on the formatting progress. This parameter can be NULL when no information is required.	
Return parameters	Type	Description	
		TRUE Success FALSE Error	
		To query extended error information, call the GetLastError() function.	



The required runtime depends on the available memory space on the storage medium. For very large storage media, the function can take several minutes.

6.1.11.32 LslGetDiskSpace

Corresponds to [LslGetDiskSpaceH\(\)](#) function with 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
    int32_t       *pResult
);
```

6.1.11.33 LslGetDiskSpaceH

Function to query information in a specific drive needed for calculation of the use and available memory.

```
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
);
```

Transfer parameters	Type	Description
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())
ccDrive	CONST CHAR	Pointer to the drive name
pBytesPerSector	uint32_t	Pointer to a variable that contains the number of bytes per sector
PSectorsPerCluster	uint32_t	Pointer to a variable that contains the number of sectors per cluster
pTotalClusters	uint32_t	Pointer to a variable for the number of clusters
pFreeCluster	uint32_t	Pointer to a variable that contains the number of free clusters

ulOption	uint32_t	Reserved				
pResult	int32_t	<p>Pointer to an INT value that is filled with the operation result</p> <table border="1" style="margin-left: 20px;"> <tr> <td style="background-color: #e0f2ff;">≥0</td><td>Success</td></tr> <tr> <td style="background-color: #e0f2ff;"><0</td><td>Negative error code</td></tr> </table>	≥0	Success	<0	Negative error code
≥0	Success					
<0	Negative error code					
Return parameters	Type	Description				
		<p>TRUE Success</p> <p>FALSE Error</p> <p>To query extended error information, call the GetLastError() function.</p>				

Requirements

Lasal32.dll: 01.01.019 and higher

LasalOS: 01.01.079 and higher

Example

```

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

Corresponds to [LslGetDriveListShortH\(\)](#) function with 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

Function to retrieve information on the available drives, as well as their properties.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetDriveListShortH(
    int32_t ocbNum,
    uint32_t *pDriveList,
    uint32_t ulSize,
    uint32_t ulOption,
    int32_t *pResult
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())	
pDriveList	uint32_t	Pointer to an unsigned long array for information on the drives	
ulSize	uint32_t	Size of the array for the drive information	
ulOption	uint32_t	Option for specifying which information to read	
pResult	int32_t	Pointer to an INT value that is filled with the operation result.	
		≥ 0 Success < 0 Negative error code	
Return parameters	Type	Description	
		TRUE Success FALSE Error	
		To query extended error information, call the GetLastError() function.	

A typical system has 26 drives (A-Z) The array should therefore have at least 26 elements, so that it can contain all possible information.

Possible values for the ulOption parameters, which can be "OR combined", are:

```
#define DRIVELIST_OPT_MOUNTED 0x01 built-in devices only.
```

```
#define DRIVELIST_OPT_MOUNTSTATE 0x02 the status of a built-in device.
  DRIVELIST_OPT_MOUNTED must be set.
```

```
#define DRIVELIST_OPT_TYPE      0x04 device type
```

```
#define DRIVELIST_OPT_DRIVER    0x08 the specified device driver.
```

The information of a respective drive is the index of the array. Index 0 is defined as drive 'A', index -1 is drive 'B', etc.

Each index contains a 32-bit value, which contains the information of the drive depending on the ulOption parameter.

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         0x00000F00
#define DRIVELIST_DRIVER_MASK     0x00070000
```

If an index is filled with 0, no drive is available or there is no information for the specified option, which can be transferred to the function with the ulOption parameter.

Requirements

Lasal32.dll: 01.01.019 and higher

LasalOS: 01.01.079 and higher

Example

```
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 LsIRenameFileDir

Corresponds to [LsIRenameFileDirH\(\)](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LsIRenameFileDir(
    const char    *ccFileDialogName,
    const char    *ccNewName,
    uint32_t      ulOption,
    int32_t       *pResult
);
```

6.1.11.37 Ls1RenameFileDirH

The Ls1RenameFileDirH() function is used to rename a file or directory.

```
extern "C" LSL_BOOL LASAL32_EXPORTS Ls1RenameFileDirH(
    int32_t          ocbNum,
    const char      *ccFileDialogName,
    const char      *ccNewName,
    uint32_t         ulOption,
    int32_t          *pResult
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online Connection Block number (return value of OcbOpen())	
ccFileDialogName	CONST CHAR	Pointer to file or directory name to be changed	
ccNewName	CONST CHAR	Pointer to the new name	
ulOption	uint32_t	Reserved	
pResult	int32_t	Pointer to an INT value that is filled with the operation result	
		≥0	Success
		<0	Negative error code
Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error
		To query extended error information, call the GetLastError() function.	

Requirements

Lasal32.dll: 01.01.019 and higher

LasalOS: 01.01.079 and higher

Example

```
BOOL rc;
int iRC;

rc = Ls1RenameFileDirH(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 Parameters of a CB_FUNCTYPE Callback Function

Transfer parameters		Type	Description
Val			Callback function is called with the number of remaining blocks as a parameter before a block is transferred
Return parameters		Type	Description
			<p>CBRETURN_CONTI Continue transfer NUE (0)</p> <p>CBRETURN_ABOR Abort transfer T (1)</p>

6.1.11.39 Parameters of a CB_FUNCTYPE2 Callback Function

Transfer parameters		Type	Description								
pData	void	Pointer pUser passed to the file transfer function (LslFileTransfer(), LslFileTransferFrom PlcBuf(), LslFileTransferToPlcBuf(), LslDownloadOSH())									
Val	uint32_t	Depending on the State parameter, various values are transferred here. See also parameter State.									
State	uint32_t	<p>Shows the current status of the file transfer. All possible states are listed in the following table.</p> <table> <tr> <td>CBSTATE_INIT</td> <td>The file transfer starts and Val shows the total number of bytes that will be transferred (file length).</td> </tr> <tr> <td>CBSTATE_STATE_BEGIN</td> <td>The file transfer starts and Val shows the total number of data blocks that will be transferred.</td> </tr> <tr> <td>CBSTATE_STATE</td> <td>The file transfer is running. Val shows the number of bytes that have already been copied.</td> </tr> <tr> <td>CBSTATE_FINISH</td> <td>The file transfer is complete. Val shows the total number of bytes that have been transferred.</td> </tr> </table>		CBSTATE_INIT	The file transfer starts and Val shows the total number of bytes that will be transferred (file length).	CBSTATE_STATE_BEGIN	The file transfer starts and Val shows the total number of data blocks that will be transferred.	CBSTATE_STATE	The file transfer is running. Val shows the number of bytes that have already been copied.	CBSTATE_FINISH	The file transfer is complete. Val shows the total number of bytes that have been transferred.
CBSTATE_INIT	The file transfer starts and Val shows the total number of bytes that will be transferred (file length).										
CBSTATE_STATE_BEGIN	The file transfer starts and Val shows the total number of data blocks that will be transferred.										
CBSTATE_STATE	The file transfer is running. Val shows the number of bytes that have already been copied.										
CBSTATE_FINISH	The file transfer is complete. Val shows the total number of bytes that have been transferred.										

		CBSTATE_ERROR An error has occurred. The Val parameter contains an error number.				
		CBSTATE_CANCEL ED The callback function has canceled the file transfer.				
Return parameters	Type	Description				
	uint32_t	<p>The callback function can return the following values:</p> <table> <tr> <td>CBRETURN_CONTINUE NUE</td><td>Continue file transfer</td></tr> <tr> <td>CBRETURN_ABORT T</td><td>Abort file transfer</td></tr> </table>	CBRETURN_CONTINUE NUE	Continue file transfer	CBRETURN_ABORT T	Abort file transfer
CBRETURN_CONTINUE NUE	Continue file transfer					
CBRETURN_ABORT T	Abort file transfer					

6.1.12 Managing the RefreshList

A refresh list is used to optimize the data exchange between a PLC and an external station (e.g. PC + Windows). With a refresh list, LASAL servers and global variables can be declared. To register object member variables, the application software itself must obtain the address. See below and chapter 8, Data exchange with object member variables. Every change in the data (values) of a declared element is then communicated to the caller (callback). This reduces the load on the communication to a minimum, which allows the user to target their reaction to a data change. There is no longer a need for a continuous query for data changes.

In the following pages, all functions for creating, managing and deleting a refresh list are explained and illustrated with short examples.

In the system (PLC), two refresh lists are available. These two lists are processed in the PLC separately and are freely available to the user. To better distinguish between the two lists, the following constants are defined.

```
#define LSL_RL_FLAG_DYNAMIC_LIST      0x00000000
#define LSL_RL_FLAG_PERMANENT_LIST     0x00000001
```

The LSL_RL_FLAG_DYNAMIC_LIST can, for example, be used for temporary values that don't need to be continuously monitored. In the LSL_RL_FLAG_PERMANENT_LIST, however, are only entries whose values are continuously required.

Definitions for the Constants and Structures Required for Data Allocation

```
#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;    // Variable or Server address
    DWORD dwType;    // input type: LSL_RL_TYPE_...
    DWORD dwSize;
};

#pragma pack(pop)
```

Create Refresh List

1. First, the refresh list must be created with `LslRefreshListCreateExt()`. This also establishes the connection to the refresh list.
2. To register numeric servers (integer and REAL), string servers and global variables (integer, REAL, LREAL, ARRAY and STRUCT) with the refresh list, the functions `LslRefreshListGetVarInfo()` and `LslRefreshListAdd()` are called for each variable. Hint: before calling `LslRefreshListAdd()`, the size of the variable in bytes must be entered in `SRLVarInfo VarInfo.dwSize`. For large arrays and structures, the parameter `dwRefreshTime` must be set to a correspondingly large value. Attention: For structures, the alignment on the PC and on the PLC must be the same.
3. To register object member variables (or memory areas allocated with `malloc()`) with the refresh list, the address of an address server must first be read with `LslRefreshListGetVarInfo(iRLB, "Objektname.AdressenServername", &VarInfo)`. The address of the object member variable or the memory area can then be read from the address server using `LslRefreshListGetData(iRLB, &VarInfo, (uint8_t*)&VarInfo.dwAddr, sizeof(plcptr_t))`. Hint: Before calling `LslRefreshListAdd()`, the following elements must be supplied in `SRLVarInfo VarInfo`: `dwAddr=determined address`, `dwSize=size of the variable in bytes`, `dwType= LSL_RL_TYPE_GLOBAL`. For large arrays and structures, the parameter `dwRefreshTime` must be set to a correspondingly large value. Attention: For structures, the alignment on the PC and on the PLC must be the same.

- When the refresh list is ready, monitoring of the variables it contains can be started with the `LslRefreshListStart()` function. From now on, the callback function specified in point 1 is called when the value of a variable in the refresh list changes.

The functions `LslRefreshListGetVarInfo()`, `LslRefreshListGetData()` and `LslRefreshListAdd()` do not require an online connection to the PLC, but only the connection to the refresh list that is established by the function `LslRefreshListCreateExt()`.

For more information on the callback function, see the prototype of the callback function in the chapter `LslRefreshListSetCallback`.

Read Data without Entry in the Refresh List

- `LslRefreshListGetVarInfo()` determines the address of the numeric server (integer or REAL) or the string server or the global variable (integer, REAL, LREAL, ARRAY and STRUCT).
- The data can then be read with `LslRefreshListGetData()` or `LslRefreshListGetDataSize()`.

Write Data without Entry in the Refresh List

- `LslRefreshListGetVarInfo()` determines the address of the numeric server (integer or REAL) or the global variable (integer, REAL, LREAL, ARRAY and STRUCT).
- The data can then be written with `LslRefreshListSetData()`.

Example of the Refresh List

```
#include "Lasal32\lsl_stdhdr.h"
#include "Lasal32\lsl_stdint.h"
#include "Lasal32\Lasal32.h"

#include <conio.h>
#include <string>
#include <thread>

//How must the values be read from the PLC
typedef enum
{
    RFRSHLIST_READTYPE_NUMSERVER,           //numerical server (Integer, REAL)
    RFRSHLIST_READTYPE_STRSERVER,           //String server
    RFRSHLIST_READTYPE_VAR,                 //All variables with 1, 2 or 4 bytes length
    RFRSHLIST_READTYPE_ARRAY,               //Arrays
    RFRSHLIST_READTYPE_STRUCT,              //Structs
    RFRSHLIST_READTYPE_LREAL,               //LREAL
} RFRSHLIST_READ_TYPE;

//How should the values be interpreted
```

```

typedef enum
{
    RFRSHLIST_TYPE_STRING_A,      //ASCII string
    RFRSHLIST_TYPE_SINT,          //Variable 1 byte signed int
    RFRSHLIST_TYPE_INT,           //Variable 2 bytes signed int
    RFRSHLIST_TYPE_DINT,          //Variable 4 bytes signed int
    RFRSHLIST_TYPE_USINT,         //Variable 1 byte unsigned int
    RFRSHLIST_TYPE_UINT,          //Variable 2 bytes unsigned int
    RFRSHLIST_TYPE_UDINT,         //Variable 4 bytes unsigned int
    RFRSHLIST_TYPE_REAL,          //Variable 4 bytes float
    RFRSHLIST_TYPE_LREAL,         //Variable 4 bytes double
    RFRSHLIST_TYPE_ARRAY_1,       //Array Type1
    RFRSHLIST_TYPE_ARRAY_2,       //Array Type2
    RFRSHLIST_TYPE_STRUCT_1,      //Structure Type1
    RFRSHLIST_TYPE_STRUCT_2,      //Structure Type2
} 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 //Address server for DataClass1.StructVar1
#define VAR_ID_DataClass1_StructVar1  3
#define VAR_ID_DataClass1_AddrStruct2 4 //Address server for DataClass1.StructVar2
#define VAR_ID_DataClass1_StructVar2  5
#define VAR_ID_DataClass1_AddrArray1  6 //Address server for DataClass1.Array1
#define VAR_ID_DataClass1_Array1     7
#define VAR_ID_DataClass1_AddrArray2  8 //Address server for DataClass1.Array2
#define VAR_ID_DataClass1_Array2     9

//Structure of a MyVarInfo[] element
typedef struct {
    long                  iRLB;
    SRLVarInfo           VarInfo;
    RFRSHLIST_READ_TYPE  RdType;    //How must the values be read from the PLC
    RFRSHLIST_DATA_TYPE  DtType;    //How should the values be interpreted
    long                  size;      //Variable length in bytes
    void*                pDest;     //Ptr to the memory location
} 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 for reading and evaluating a structure
 //*****
 void ReadStruct(VarInfoStruct* pVarInfo, uint32_t dwAddr)
 {
    StructType1* pStructVar1;
    StructType2* pStructVar2;

    if (LslRefreshListGetData(pVarInfo->iRLB, &pVarInfo->VarInfo, (uint8_t*)pVarInfo->pDest, p
        {
            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, pStructVar2->iValue3);
                }
                break;
            }
        }
        else
            printf("LslRefreshListGetData() failed(0x%x)\n", GetLastError());
    }

    //*****
    //Thread for reading and evaluating a string server
    //*****
    void ReadString(VarInfoStruct* pVarInfo, uint32_t dwAddr)
    {
        uint32_t iTrueLenght = 0;

        if (LslRefreshListGetDataSize(pVarInfo->iRLB, &pVarInfo->VarInfo, &iTrueLenght, (uint8_t*)pVarInfo->pDest)
            printf("String server String=%s\n", (char*)pVarInfo->pDest);
        else
            printf("LslRefreshListGetDataSize() failed(0x%x)\n", GetLastError());
    }

    //*****
    //Thread for reading and evaluating an array
    //*****
    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, CallbackFunc)  
    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*)&pVarInfo
    {
        //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_StructVar1
        {
            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*)&pVarInfo
    {
        //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_StructVar2
        {
            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", &pVarInfoAdd
{
    //Read the address of the array from the server "AddrArray1"
    pVarInfo = &MyVarInfo[VAR_ID_DataClass1_Array1];
    if (LslRefreshListGetData(iRefreshList, &pVarInfoAddr->VarInfo, (uint8_t*)&pVa
    {
        //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_Ar
        {
            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", &pVarInfoAdd
{
    //Read the address of the array from the server "AddrArray2"
    VarInfo = &MyVarInfo[VAR_ID_DataClass1_Array2];
    if (LslRefreshListGetData(iRefreshList, &pVarInfoAddr->VarInfo, (uint8_t*)&pVa
    {
        //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_Ar
        {
            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].VarInfo);

    //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, (uint8_t*)&dwnewValue,
    //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, (uint8_t*)&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

This function adds a new entry to the end of the refresh list. The variable information can be determined with the [LslRefreshListGetVarInfo\(\)](#) function. When selecting the refresh time (dwRefreshTime parameter), how often the value changes and/or in which time period it is monitored by the PLC should be taken into consideration. Because the human eye is very slow and each display glows, a value on the screen with a refresh rate of less than 50 ms is unpleasant.

A rule of thumb

250 ms	"slow" values
100 ms	"standard" values
50 ms	"fast" values

It is important to note which of the two refresh lists are addressed with the dwFlag parameter.



```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListAdd(int iRLB, const SRLVarInfo *pVarInfo, ...)
```

Transfer parameters	Type	Description				
iRLB	INT	Refresh list ID				
pVarInfo	const SRLVarInfo	Pointer to SRLVarInfo (e.g. created by LslRefreshListGetVarInfo())				
dwVarID	DWORD	User-specific ID to clearly identify this entry				
dwRefreshTime	DWORD	Refresh time for this variable; minimum time setting is 10 ms				
dwFlag	DWORD	LSL_RL_FLAG_DYNAMIC_LIST or LSL_RL_FLAG_PERMANENT_LIST or LSL_RL_FLAGADD_NOANSWER (Does not wait for an answer after a new variable was added. It is faster but does not test whether the variable was added correctly)				
Return parameters	Type	Description				
		<table border="1"> <tr> <td>TRUE</td><td>Success</td></tr> <tr> <td>FALSE</td><td>Error</td></tr> </table> <p>To retrieve the error information, use the Windows function GetLastError().</p>	TRUE	Success	FALSE	Error
TRUE	Success					
FALSE	Error					

6.1.12.2 LslRefreshListClear

This function deletes all entries from the refresh list. The RefreshList ID is however, still valid and the callback function is also still entered.

It is important to note which of the two refresh lists are addressed.



```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListClear(int iRLB, DWORD dwFlag);
```

Transfer parameters		Type	Description	
iRLB		INT	Refresh list ID	
dwFlag		DWORD	LSL_RL_FLAG_DYNAMIC_LIST or LSL_RL_FLAG_PERMANENT_LIST	
Return parameters		Type	Description	
			TRUE	Success
			FALSE	Error
			To retrieve the error information, use the Windows function GetLastError() .	

6.1.12.3 LslRefreshListCreate

This function creates a new refresh list. In the event of an error, the return value is 0. Otherwise a valid ID is returned. This ID is needed each time the program accesses the refresh list.

```
extern "C" int LASAL32_EXPORTS LslRefreshListCreate();
```

Transfer parameters		Type	Description	
none				
Return parameters		Type	Description	
			0	Error
			#0	Refresh list ID

6.1.12.4 [LslRefreshListCreateExt](#)

Combines several functions ([LslRefreshListCreate\(\)](#), [LslRefreshListOnline\(\)](#), [LslRefreshListSetCallback\(\)](#), [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
);
```

Transfer parameters	Type	Description				
szComm	const char*	ASCII-0 string that specifies the interface used to establish a connection Example: TCP:10.10.116.42				
uBaudRate	uint8_t	0; currently not supported				
uPcStation	uint8_t	0; currently not supported				
uSpsStation	uint8_t	0; currently not supported				
lpcName	const char*	Object name of the symbol table				
pCallback	CB_RLADD_FUNCT YPE *	Pointer to the current callback function. The protoType of this function is CB_RLADD_FUNCType (see below).				
pCallbackData	void*	Void pointer that is loaded with each call of the callback function				
dwTimeoutMS	DWORD	If the entered value is unlike 0, the command then checks whether it is possible to go online within the specified time and whether the control is in RunRAM or RunROM before creating the RefreshList.				
Return parameters	Type	Description				
	INT	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20px; text-align: center;">0</td><td>Error</td></tr> <tr> <td style="text-align: center;">≠0</td><td>ID of the defined RefreshList</td></tr> </table>	0	Error	≠0	ID of the defined RefreshList
0	Error					
≠0	ID of the defined RefreshList					

Example

See behavior of the refresh list.

6.1.12.5 LslRefreshListDestroy

This function deletes an existing refresh list. The parameter is the ID of the refresh list to be deleted. Normally the return value is 0, if an error occurs, a non-zero value is returned. An existing connection to the PLC is naturally deleted. The refresh list ID can then no longer be used.



This function returns both a zero and non-zero value. Some functions return TRUE or FALSE.

```
extern "C" int LASAL32_EXPORTS LslRefreshListDestroy(int iRLB);
```

Transfer parameters		Type	Description	
iRLB		INT	Refresh list ID	
Return parameters		Type	Description	
			0	Success
			#0	Error
With GetLastError() , the last error code can be determined. This code corresponds to either a Lasal32 error code or a system-specific error (Windows: GetLastError() , or Linux: errno).				

6.1.12.6 LslRefreshListGetData

With this function, the value of a variable or server can be read without an entry in RefreshList.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListGetData(int iRLB, const SRLVarInfo *pVarInfo
```

Transfer parameters		Type	Description	
iRLB		INT	Refresh list ID	
pVarInfo		CONST SRLVarInfo	Pointer to the initialized SRLVarInfo structure	
pData		BYTE	Pointer to a byte buffer (destination)	
dwSize		DWORD	Buffer size in bytes	
Return parameters		Type	Description	
			TRUE	Success

		FALSE Error
To query extended error information, call the GetLastError() function.		

Example 1 (read string server)

```
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);
}
```

Example 2 (read numerical server)

```
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);

```

Example 3 (read global variable)

```

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

This function can be used to read the value and length of a variable or a server. No entry in the refresh list is required for this. With a string server, the correct string length is

returned in pSize and the 0-terminated string is returned in the pData buffer if the buffer is large enough.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListGetDataSize(
    int32_t           iRLB,
    const SRLVarInfo* pVarInfo,
    DWORD*            pSize,
    unsigned char*    pData,
    DWORD             dwSize
);
```

Transfer parameters	Type	Description	
iRLB	int32_t	ID of the refresh list (result of LslRefreshListCreate() or LslRefreshListCreateExt())	
pVarInfo	const SRLVarInfo*	Pointer to SRLVarInfo (created by LslRefreshListGetVarInfo())	
pSize	DWORD*	Pointer to a variable where the string length without 0 terminator is entered for a string server	
pData	unsigned char*	Pointer to the result buffer	
dwSize	DWORD	Size of pData in bytes. Must be one greater than the string length displayed in pSize for the 0 terminator.	
Return parameters	Type	Description	
		TRUE	Fuction successful
		FALSE	Error
		To obtain extended error information, call the GetLastError() function.	

Example

This allows you to read a string optimized as follows. This procedure has the advantage that you do not have to make a network communication again for smaller strings, as the string has already been read and is contained in buf. Only if this is too small do you have to read again with a larger buffer.

```
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);
            str = std::string(ptr, size_string);
            free(ptr);
        } else {
            str = std::string(buf, size_string);
        }
    }
}
```

```

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

Returns the version number of the loader.

```
extern "C" DWORD LASAL32_EXPORTS LslRefreshListGetLoaderVersion(
    int32_t    iRLB
);
```

Transfer parameters		Type	Description				
iRLB		int32_t	ID of the refresh list (result of LslRefreshListCreate() or LslRefreshListCreateExt())				
Return parameters		Type	Description				
			<table border="1"> <tr> <td>>0</td><td>Loader version number</td></tr> <tr> <td>0</td><td>Error</td></tr> </table> <p>To obtain extended error information, call the GetLastError() function.</p>	>0	Loader version number	0	Error
>0	Loader version number						
0	Error						

The loader version number has the following structure:

HIGH-WORD				LOW-WORD			
31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
				Major	Minor	Subversion	

Example

The loader version V02.02.85 corresponds to the hex value 16#00002255.

6.1.12.9 LslRefreshListGetVarInfo

This function determines the required information on the variable. Currently, only servers with the notation Objectname.servername and global variables (simple variables such as INT, SINT, DINT, UINT, USINT, UDINT, REAL, LREAL as well as structures and arrays) are supported. Only the address and type are determined for all global variables, but no length. The length must be subsequently entered in the SRLVarInfo structure by the caller.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListGetVarInfo(int iRLB, const char *lpcName, ...);
```

Transfer parameters		Type	Description	
iRLB		INT	Refresh list ID	
lpcName		CONST CHAR	Name of the server or variable (ASCII-0 string)	
pVarInfo		SRLVarInfo	Pointer to SRLVarInfo; the structure is initialized using this function	
Return parameters		Type	Description	
			TRUE	Success
			FALSE	Error
			To query extended error information, call the GetLastError() function.	

6.1.12.10 LslRefreshListIsOnline

This function checks for an existing Online connection. If a valid connection is found, TRUE is returned. If no valid connection is available, the return value is false.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListIsOnline(int iRLB);
```

Transfer parameters		Type	Description	
iRLB		INT	Refresh list ID	
Return parameters		Type	Description	
			TRUE	Success

		FALSE Error
To query extended error information, call the GetLastError() function.		

6.1.12.11 LslRefreshListOffline

This function ends the connection with the PLC and thereby stops the data exchange between the PLC and the external station. All entries are then removed from the refresh list but the actual refresh list is not deleted and the pointer to the callback function remains. The refresh ID therefore remains valid and can be used further.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListOffline(int iRLB);
```

Transfer parameters	Type	Description
iRLB	INT	Refresh list ID
Return parameters	Type	Description
		TRUE Success
		FALSE Error
To query extended error information, call the GetLastError() function.		

6.1.12.12 LslRefreshListOnline

This function is used to establish a connection to the PLC. The parameters needed for this function are the refresh list ID and the connection data



Currently only the TCP connection is supported.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListOnline(int iRLB, const char* szComm, BYTE uB
```

Transfer parameters	Type	Description
iRLB	INT	Refresh list ID
szComm	CONST CHAR	ASCII-0 string that specifies the interface used to establish a connection.

		Example: TCP:10.10.116.42
uBaudRate	BYTE	0, currently not supported
uPcStation	BYTE	0, currently not supported
uSpsStation	BYTE	0, currently not supported
Return parameters	Type	Description
	LSL_BOOL	<div style="display: flex; justify-content: space-around; align-items: center;"> TRUE Success </div> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 10px;"> FALSE Error </div> <p>To query extended error information, call the GetLastError() function.</p>

6.1.12.13 LslRefreshListSetCallback

With this connection, the callback function for the refresh list is initialized. The callback function is for the actual data transfer; it is called each time the data in the refresh list is changed.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListSetCallback(int iRLB, CB_RLADD_FUNCTYPE *p
```

Transfer parameters	Type	Description
iRLB	INT	Refresh list ID
pCallback	CB_RLADD_FUNCTYPE	Pointer to the actual CallBack function. The protoType of this function is CB_RLADD_FUNCType (see below).
pCallbackData	VOID	Void pointer that is given each time the callback function is called
Return parameters	Type	Description
		<div style="display: flex; justify-content: space-around; align-items: center;"> TRUE Success </div> <div style="display: flex; justify-content: space-around; align-items: center; margin-top: 10px;"> FALSE Error </div> <p>To query extended error information, call the GetLastError() function.</p>

CallBack Function Prototype

```
void _cdecl RTRLCounterCallback(void *pCallbackData, DWORD dwAddr, DWORD dwVarID, int dNewDa
```

Transfer parameters	Type	Description
----------------------------	-------------	--------------------

pCallbackData	VOID	User-defined void pointer (see above)
dwAddr	DWORD	Defined address managed by the system
dwVarID	DWORD	User-specific ID of the entry in RefreshList (see function LslRefreshListAdd())
dNewData	INT	New value in the PLC

After the refresh list is started, the system is responsible for calling the callback function. The user need only accept the changed data and if necessary, start additional actions in regard to the application. After the refresh list is stopped, this function is no longer called. It should be noted that the callback function is processed as quickly as possible. This means that the program code contained within should be as short as possible.

The callback should contain no further calls from the Lasal32.dll function.



```
typedef void __cdecl CB_RLADD_FUNCTYPE (
void * pCallbackData,
DWORD dwAddr,
DWORD dwVarID,
int dNewData);
```

Notes

1. If the value of a numeric server (integer or floating point number REAL) changes, the new value is passed to the callback function in the parameter dNewData.
2. If the value of a string server changes, the new value must be read from the string server. The `LslRefreshListGetDataSize()` function is well suited for this purpose. However, the `LslRefreshListGetData()` function can also be used. These functions must be called from a separate thread.
3. If the value of a global variable changes, the new value must be read from the PLC. The `LslRefreshListGetData()` function is well suited for this purpose. However, the `LslRefreshListGetDataSize()` function can also be used. These functions must be called from a separate thread.
4. Object member variables or memory areas allocated with `malloc()` were declared as `LSL_RL_TYPE_GLOBAL` and are therefore to be treated as described in point 3.

The functions `LslRefreshListGetData()` and `LslRefreshListGetDataSize()` do not require an online connection to the PLC, but only the online connection to the refresh list that is established by the function `LslRefreshListCreateExt()`.

6.1.12.14 LslRefreshListSetData

This function writes data to a variable or server. Whether the server (or variable) is entered in RefreshList is irrelevant.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListSetData(int iRLB, const SRLVarInfo *pVarIn
```

Transfer parameters	Type	Description	
iRLB	INT	Refresh list ID	
pVarInfo	CONST SRLVarInfo	Pointer to the initialized SRLVarInfo structure	
pData	CONST BYTE	Data address (source)	
dwSize	DWORD	Data size in bytes	
Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error
To query extended error information, call the GetLastError() function.			

Example (C/C++)

```
// Global counter
DWORD g_dwCounterSpeedTest = 0;

// Callback function (increments the global counter)
void __cdecl RTRLCounterCallback(
void *pCallbackData,
DWORD dwAddr,
DWORD dwVarID,
int dNewData);

{
    g_dwCounterSpeedTest++;
}

// Test function
void TestRefreshListSpeed(int iRLB)
{
    g_dwCounterSpeedTest = 0;
    DWORD dwStart;

    SRLVarInfo varInfoStart, varInfoCounter;

    // determine Info for the g_RTRFStart variable
    if (LslRefreshListGetVarInfo(iRLB, "g_RTRFStart", &varInfoStart) == TRUE)
```

```
{  
    // determine server information.  
    if (LslRefreshListGetVarInfo(iRLB, "ClassRealTimeRefreshList1.SrvCounter", &varInfoCounter  
    {  
        DWORD dwWait = 3000;  
  
        printf("RealTime ReaefreshList Counter Test is running !\n");  
  
        // Set the Callback function  
        LslRefreshListSetCallback(iRLB, RTRLCounterCallback, NULL);  
        // Add a variable to RefreshListe.  
        LslRefreshListAdd(iRLB, &varInfoCounter, 1234, 100, 0);  
        // Start RefreshListe  
        LslRefreshListStart(iRLB, 0);  
  
        Sleep(10);  
  
        dwStart = 1;  
        // Start the internal counter in the control.  
        LslRefreshListSetData(iRLB, &varInfoStart, (const BYTE *) &dwStart,  
        sizeof(dwStart));  
  
        // Wait a specified amount of time.  
        // within this period, the Callback function is called.  
        Sleep(dwWait);  
  
        dwStart = 0;  
        // Stop the internal counter in the control.  
        LslRefreshListSetData(iRLB, &varInfoStart, (const BYTE *) &dwStart,  
        sizeof(dwStart));  
  
        // Delete the RefreshListe  
        LslRefreshListClear(iRLB, 0);  
        // Display the measurement results.  
        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()  
{  
    // Generate a RefreshListBlock.  
    int iRLB = LslRefreshListCreate();  
  
    // Use the TCP in the control to go Online  
    if (LslRefreshListOnline(iRLB, "TCP:10.10.116.43", 0, 0, 0))  
    {  
        // Check whether an Online connection has been established.  
        if (LslRefreshListIsOnline(iRLB))
```

```

    printf("Online ok\n");
else
    printf("Online NOT ok\n");

// Run test function.
TestRefreshListSpeed(iRLB);

// Go offline
LslRefreshListOffline(iRLB);
}
else
printf("Online error 0x%08X\n", GetLastError() );

// Delete RefreshListBlock.
LslRefreshListDestroy(iRLB);
}

```

For this example, the corresponding application must run in the control!

6.1.12.15 LslRefreshListSymbolTableInit

```
extern "C" uint32_t LslRefreshListSymbolTableInit(int iRLB, const char *lpcName);
```

Transfer parameters	Type	Description
iRLB	INT	Refresh list ID
lpcName	CONST CHAR	Object name of the symbol table
Return parameters	Type	Description
		If an error occurs, RefreshList ID = 0 is returned, otherwise the return value is zero

6.1.12.16 LslRefreshListStartCount

With this function, the number of entries in a refresh list can be controlled. With dwCount, only the actual number of entries are transferred. If the number of entries given is 0, this function operates like [LslRefreshListClear\(\)](#).

It is important to note which of the two refresh lists are addressed.



```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListStartCount(int iRLB, DWORD dwCount, DWORD dwCount)
```

Transfer parameters		Type	Description
iRLB		INT	Refresh list ID
dwCount		DWORD	Number of valid entries in RefreshList
dwFlag		DWORD	LSL_RL_FLAG_DYNAMIC_LIST or LSL_RL_FLAG_PERMANENT_LIST
Return parameters		Type	Description
			<p>TRUE Success</p> <p>FALSE Error</p> <p>To query extended error information, call the GetLastError() function.</p>

6.1.12.17 LslRefreshListStart

This function starts the actual refresh for all entries. A one-time call of this function is required after successfully adding entries with the [LslRefreshListAdd\(\)](#) function. A one-time call to the entire RefreshList is also started after several calls of the [LslRefreshListAdd\(\)](#) function.

It is important to note which of the two refresh lists are addressed.



```
extern "C" LSL_BOOL LASAL32_EXPORTS LslRefreshListStart(int iRLB, DWORD dwFlag);
```

Transfer parameters		Type	Description
iRLB		INT	Refresh list ID
dwFlag		DWORD	LSL_RL_FLAG_DYNAMIC_LIST or LSL_RL_FLAG_PERMANENT_LIST
Return parameters		Type	Description
			<p>TRUE Success</p> <p>FALSE Error</p> <p>To query extended error information, call the GetLastError() function.</p>

6.1.12.18 Ls1SetDbgLevelEx

Changes the debug level of lasal32.dll or the refresh list to the specified value if the new level is ≥ 0 . The return value is the old debug level.

```
extern "C" int32 LASAL32_EXPORTS Ls1SetDbgLevelEx(
    int32_t    level,
    uint32_t   dwFlags
);
```

Transfer parameters	Type	Description
level	int32_t	New debug level ≥ 0
dwFlags	uint32_t	DBGLEVELLOG_LASAL32: change debug level of lasal32.dll DBGLEVELLOG_REFRESHLIST: change debug level of the refresh list Attention: Only one flag may be set at a time. If both flags are set, then only the debug level of lasal32.dll is changed.
Return parameters	Type	Description
		Old debug level

6.1.13 Internal Functions

6.1.13.1 Ls1DownloadOS

Corresponds to [Ls1DownloadOS](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS Ls1ReadFromClt(
    uint32_t  objAddr,
    uint32_t  *dValue
);
```

6.1.13.2 Ls1DownloadOSH

This function installs the specified operating system on the PLC.

```
typedef uint32_t __cdecl CB_FUNCTYPE2(void *, uint32_t val, uint32_t state);

extern "C" LSL_BOOL LASAL32_EXPORTS Ls1DownloadOSH(
    int32_t      ocbNum,
    const char   *fileSrc,
    CB_FUNCTYPE2 *callback,
    void         *pUser
);
```

Transfer parameters	Type	Description
---------------------	------	-------------

ocbNum	int32_t	Online Connection Block number (return value of OcbOpen)
fileSrc	CONST CHAR	File name of the new OS download file
Callback	CB_FUNCTYPE2*	<p>Pointer to a CB_FUNCTYPE2 callback function. When transferring the file, it is possible to divide the data into blocks. Before the file is transferred, the callback function is called once with the file length in bytes and once with the number of remaining blocks as parameters. The callback function is called during the transfer of the file with the number of bytes that have already been copied as a parameter.</p> <p>The callback function can be used, for example, to update a progress bar. If the callback parameter is NULL, no callback function is called.</p>
pUser	VOID*	User parameter sent to the callback function

6.1.14 LsIDirect

With the LsIDirect commands, it is possible to establish a simple connection to the LsIDirect class in the PLC via TCP / IP. Up to 100 online connections (Direct Online Blocks (DOBS)) are possible.

6.1.14.1 LsIDirectCreateDOB

Applies a DOB and returns the index of the block when successful.

```
extern "C" int32_t LsIDirectCreateDOB();
```

Transfer parameters	Type	Description				
none						
Return parameters	Type	Description				
		<table border="1"> <tr> <td>≥0</td> <td>Valid DOB</td> </tr> <tr> <td>-1</td> <td>An error has occurred (use GetLastError() function)</td> </tr> </table>	≥0	Valid DOB	-1	An error has occurred (use GetLastError() function)
≥0	Valid DOB					
-1	An error has occurred (use GetLastError() function)					

6.1.14.2 LsIDirectDestroyDOB

Frees the resources previously occupied by [LsIDirectCreateDOB\(\)](#). The connection is then closed, if not already.

```
extern "C" int32_t LsIDirectDestroyDOB(int32_t nDOB);
```

Transfer parameters	Type	Description	
nDOB	int32_t	nDOB from LsIDirectCreateDOB()	
Return parameters	Type	Description	
		0	Success
		#0	Error code

6.1.14.3 LsIDirectIsOnline

Checks whether the resource transferred with nDOB is still connected.

```
extern "C" LSL_BOOL LsIDirectIsOnline(int32_t nDOB);
```

Transfer parameters	Type	Description	
nDOB	int32_t	Valid DOB from LsIDirectCreateDOB()	
Return parameters	Type	Description	
		FALSE	Not online or an error has occurred
		TRUE	Online OK

6.1.14.4 LsIDirectOffline

Ends the connection made with [LsIDirectOnline\(\)](#). (Here, the resource is not freed. To free the occupied resource, [LsIDirectDestroyDOB](#) must be called.

```
extern "C" int32_t LsIDirectOffline(int32_t nDOB);
```

Transfer parameters	Type	Description	
nDOB	int32_t	Valid DOB from LsIDirectCreateDOB()	
Return parameters	Type	Description	
		0	Success
		#0	Error code

6.1.14.5 LslDirectOnline

Establishes a connection with a TCP/IP. If pCallback is not NULL, this function is called when data was received. Otherwise, see [LslDirectReceive\(\)](#).

```
extern "C" int32_t LslDirectOnline(
    int32_t             nDOB,
    const char          *lpTcp,
    uint32_t             nPort,
    DO_CALLBACK_FUNCTYPE *pCallback,
    Void                *pCookie
);
```

Transfer parameters		Type	Description	
nDOB		int32_t	nDOB from LslDirectCreateDOB()	
lpTcp		const char	Format of the string TCP:10.10.115.43	
nPort		uint32_t	If nPort is 0, the default port is used	
pCallback		DO_CALLBACK_FUNCTYPE	Pointer to a callback function	
pCookie		Void	User data transferred with the callback	
Return parameters		Type	Description	
			≥0	Valid DOB
			-1	Error

6.1.14.6 LslDirectReceive

Returns the data received from the buffer.

```
extern "C" int32_t LslDirectReceive(int32_t nDOB, uint32_t *pnCmd, uint32_t *pnRealSize, uint8_t
```

```
#define LSL_DO_RECEIVE_NOWAIT 0x00000001
// Returns ERROR_SIGMA32_DO_RECEIVE_NO_DATA when no data is available
```

```
#define LSL_DO_RECEIVE_NOTIMEOUT 0x00000002
// Wait for data with no timeout (default 10 s)
```

Transfer parameters		Type	Description	
nDOB		int32_t	Valid DOB from LslDirectCreateDOB()	
pnCmd		uint32_t	Command	
pnRealSize		uint32_t	Length of the data received	

pBuffer	uint8_t	Buffer in which the data will be stored	
nBufSize	uint32_t	Data buffer size	
dwFlags	uint32_t	See #define	
Return parameters	Type	Description	
		0	Success
		≠0	Error code

6.1.14.7 LsIDirectReceiveCount

Returns the number of data packets found in the buffer.

```
extern "C" int32_t LsIDirectReceiveCount(int32_t nDOB, uint32_t *pCount);
```

Transfer parameters	Type	Description	
bDOB	int32_t	Valid DOB from LsIDirectCreateDOB()	
pCount	uint32_t	Data packet counter	
Return parameters	Type	Description	
		0	Success
		≠0	Error code

6.1.14.8 LsIDirectSend

Sends a command to the target with the required data in pSend.

```
extern "C" int32_t LsIDirectSend(int32_t nDOB, uint32_t nCmd, const uint8_t *pSend, uint32_t
```

Transfer parameters	Type	Description	
nDOB	int32_t	Valid DOB from LsIDirectCreateDOB	
nCmd	uint32_t	Send command ID	
pSend	const uint8_t	Pointer to the send data	
nSendLen	uint32_t	Length of the send data	
Return parameters	Type	Description	

		0 Success
		#0 Error code

6.1.14.9 LslDirectSetParam

Can be used to set the timeout for [LslDirectReceive\(\)](#).

```
extern "C" int32_t LslDirectSetParam(int32_t nDOB, uint32_t nType, uint32_t nData);
#define LSL_DO_SET_TYPE_TIMEOUT_RECEIVE 0
// data  $\mu$ s time out value in milliseconds.
```

Transfer parameters		Type	Description
nDOB		int32_t	Valid DOB from LslDirectCreateDOB()
nType		uint32_t	Parameter type (see LSL_DO_SET_TYPE_)
nData		uint32_t	Parameter data
Return parameters		Type	Description
			0 Success
			#0 Error code

6.1.15 Appendix 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	OPSY PROGRAM
28	OPSY INSTALLED
29	OPSY 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](#)

Corresponds to [LslExecNewInstrH](#) function with 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
);
```

Example

The LASAL project delivers a NewInst instruction call, with which data can be read from a WORD (UINT).

The command is configured as follows:

Command Byte	1
--------------	---

Parameter 0	startIndex in Array
Parameter 1	Number of bytes to read

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;
    Number    : 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^.uIcmd 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$UINT*Sizeof (INT); // not like 125 entries anymore
            _memcpy(#pResult^.aData, pData+ (ReadIndex * sizeof (INT)),
                    number$UDINT* sizeof (INT));
        ELSE
            ret_code:= ERROR;
        END_IF;

    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
// 
Parameters:
//  readIndex    .. Index (on an entry) in the trend buffer
Number 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 number,
                  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 - invalid Parameter !\n");
        return -1;
    }

    paras[0] = readIndex;
    paras[1] = anzahl;
```

```

if (!LslExecNewInstr(addr,
                     1,                               // wCmd,
                     paras,                          // adwParas
                     sizeof(paras),                 // dwParaCnt
                     cmdBuf,                         // dwBuf,
                     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: invalid length field received!\n");
    returnedBytes = 0;
    return -1;
}
return (returnedBytes / sizeof(WORD));
}

```

6.1.15.2 LslExecNewInstrH

Calls the NewInstr (= command call) method of an object.

```

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
);

```

Transfer parameters	Type	Description
ocbNum	int32_t	Online connection block number (return value of OcbOpen())
dwSrvAddress	uint32_t	Server address

wCmd	uint16_t	Command
adwParas	uint32_t	Command parameter (array with 4-byte value).
dwParaCnt	uint32_t	Number of command parameters. A maximum of 20 parameters are possible.
pResult	uint8_t*	Result buffer. The first 2 bytes contain the total length of the result buffer (incl. length field). The maximum length is 252 bytes.
dwResultCnt	uint32_t	Size of the result buffer in bytes
useLenField	uint32_t	This parameter is ≠ 0 when the length field in the return message should be interpreted and only the amount of bytes defined therein should be copied into the result buffer. If this parameter is 0, dwResultCnt is used for the number of bytes to copy.
Return parameters	Type	Description
		TRUE Success
		FALSE Error

With the instruction call, objects commands can be triggered. The commands are run in the NewInst method of the class. Instruction calls are often used for data transfer.

6.1.15.3 LslGetObjCls

Corresponds to [LslGetObjClsH\(\)](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjCls(
    const uint32_t objAddr,
    char           *className
);
```

6.1.15.4 LslGetObjClsH

Determines the class name of an object.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjClsH(
    int32_t        ocbNum,
    const uint32_t objAddr,
    char           *className
);
```

Transfer parameters	Type	Description
ocbNum	int32_t	Online connection block number (return value of OcbOpen())
objAddr	const uint32_t	Object address

className	char	Name of class
Return parameters	Type	Description
		TRUE Success
		FALSE Error

Example

```

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

Corresponds to [LslGetObjectH\(\)](#) function with 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

Corresponds to [LslGetObjectExH\(\)](#) with 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

This function determines an object or channel address (channel is a client or server) and also the name of the class for objects of type _CH_OBJ.

The LslGetAdressVar() function can be used to determine the addresses of global variables.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjectExH(
    int32_t      ocbNum,
    const char   *objName,
    plcptr_t     *objAddr,
    ChMode       *pMode,
    char         *pszClsName,
    uint8_t      *pDataBufferFlag
);
```

Transfer parameters		Type	Description				
ocbNum		int32_t	Online connection block number (return value of OcbOpen())				
objName		const char*	Name of the object or channel (ObjectName.ServerName or ObjectName.ClientName) as a 0-terminated ASCII string				
objAddr		plcptr_t*	Pointer to a variable where the object or channel address is entered				
pMode		ChMode*	Pointer to a variable where the channel type is entered. Values see ChMode in Lasal32.h				
pszClsName		char*	Pointer to a buffer where the name of the class is entered if it is an object of type (_CH_OBJ) (0-terminated ASCII string). This parameter may be NULL if the class name is not required.				
pDataBufferFlag		uint8_t*	Pointer to a variable where it is entered whether this channel has a data buffer (DataBufferFlag=4) or not (DataBufferFlag=0). For a channel with a data buffer (string server), more than 4 bytes of data are transferred when this channel is added to the refresh list. This parameter may be NULL if the information is not required.				
Return parameters		Type	Description				
			<table border="1"> <tr> <td>TRUE</td><td>Fuction successful</td></tr> <tr> <td>FALSE</td><td>Error</td></tr> </table> <p>To obtain extended error information, call the GetLastError() function.</p>	TRUE	Fuction successful	FALSE	Error
TRUE	Fuction successful						
FALSE	Error						

Example

```

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
{
    //Valid address in dwAddr
}
else
    printf("LslGetObjectEx failed(0x%08X)\n", GetLastError());

```

6.1.15.8 LslGetObjectH

This function determines an object or channel address (channel is a client or server) and, for objects of type `_CH_OBJ`, also the name of the class.

The `LslGetAdressVar()` function can be used to determine the addresses of global variables.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjectH(
    int32_t      ocbNum,
    const char*  *objName,
    uint32_t      *objAddr,
    ChMode       *pMode,
    char         *pszClsName = NULL
);

```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online connection block number (return value of OcbOpen())	
objName	const char*	Name of the object or channel (ObjectName.ServerName or ObjectName.ClientName) as 0-terminated ASCII string	
objAddr	uint32_t*	Object or channel address	
pMode	ChMode*	Channel type; values see ChMode in Lasal32.h	
pszClsName	CHAR*	Name of the class when it is an object of the type <code>_CH_OBJ</code>	
Return parameters	Type	Description	
		TRUE	Success
		FALSE	Error

Example

Values see ChMode in Lasal32.h.

See LslGetObjCls().

6.1.15.9 LslGetObjectID

Corresponds to LslGetObjectIDH() with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjectID(
    const char    *name,
    uint32_t      *adresse
);
```

6.1.15.10 LslGetObjectIDH

This function determines the address of the specified LASAL object.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslGetObjectIDH(
    int32_t      ocbNum,
    const char   *name,
    uint32_t      *adresse
);
```

Transfer parameters		Type	Description
ocbNum		int32_t	Online connection block number (return value of OcbOpen())
Name		const char*	Pointer to the object name (ASCII 0 string)
address		uint32_t*	Pointer to a variable where the object address is entered
Return parameters		Type	Description
			TRUE Function successful FALSE Error
			To obtain extended error information, call the GetLastError() function.

Example

```
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

Corresponds to [LslReadFromCltH\(\)](#) function with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromClt(
    uint32_t    objAddr,
    uint32_t    *dValue
);
```

6.1.15.12 LslReadFromCltH

This method calls the read method of a server that is connected to a client. The address of the client must be entered and the value read is then written to the server.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromCltH(
    int32_t    ocbNum,
    uint32_t    objAddr,
    uint32_t    *dValue
);
```

Transfer parameters	Type	Description	
ocbNum	int32_t	Online connection block number (return value of OcbOpen())	
objAddr	uint32_t	Channel address of the client	
dValue	uint32_t*	Pointer to a variable where the return value of Read method is stored	
Return parameters	Type	Description	
	LSL_BOOL	TRUE	Success
		FALSE	Error

Example

```
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

Corresponds to [LslReadFromSvrH\(\)](#) function with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromSvr(
    uint32_t    objAddr,
    uint32_t    *dValue
);

```

6.1.15.14 LslReadFromSvrH

Calls the read method of a server. This function can only be used for numerical servers.
For string servers use the function [LslReadFromSvrStrH\(\)](#).

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromSvrH(
    int32_t      ocbNum,
    uint32_t    objAddr,
    uint32_t    *dValue
);

```

Transfer parameters		Type	Description	
ocbNum		int32_t	Online connection block number (return value of OcbOpen())	
objAddr		uint32_t	Server channel address	
dValue		uint32_t	Return value of Read() method	
Return parameters		Type	Description	
			TRUE	Success
			FALSE	Error

Example

```

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

Corresponds to LslReadFromSvrStrH() with 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

This function calls the NewInst method of a string server and returns the string and the string length.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslReadFromSvrStrH(
    int32_t    ocbNum,
    plcptr_t    objAddr,
    char        *pString,
    uint32_t    dwLen,
    uint32_t    *pStringLen
);
```

Transfer parameters		Type	Description				
ocbNum		int32_t	Online connection block number (return value of OcbOpen())				
objAddr		plcptr_t	Server channel address				
pString		char*	Pointer to a buffer where the 0-terminated ASCII or Unicode string is entered				
dwLen		uint32_t	Buffer size (in bytes). Due to the zero termination, the buffer must be one byte (ASCII) or two bytes (Unicode) longer than the string length.				
pStringLen		uint32_t*	Pointer to a variable where the string length is entered in bytes (without null termination). This pointer may be NULL if the length is not required.				
Return parameters		Type	Description				
			<table border="1"> <tr> <td>TRUE</td><td>Fuction successful</td></tr> <tr> <td>FALSE</td><td>Error</td></tr> </table> <p>To obtain extended error information, call the GetLastError() function.</p>	TRUE	Fuction successful	FALSE	Error
TRUE	Fuction successful						
FALSE	Error						

Example

```

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

Corresponds to [LslWriteToCltH\(\)](#) function with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToClt(
    uint32_t    objAddr,
    uint32_t    dValue
);

```

6.1.15.18 LslWriteToCltH

This method calls the Write() method of a server that is connected with a client. The address of the client must be entered and the value of the server is changed.

```

extern "C" bool LslWriteToClt(
    DWORD    objAddr,
    DWORD    dValue
);

```

Transfer parameters	Type	Description
objAddr	DWORD	Channel address of the client

dValue	DWORD	Value that should be written to the client
Return parameters	Type	Description
		TRUE Success
		FALSE Error

Example

```

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");
            }
        }
    }
}

```

```

        printf("LslGetObject() failed 0x%08X\n", GetLastError());
    }
}
else
    printf("Error Online:0x%08X \n", GetLastError());
    ::Offline();
}

```

6.1.15.19 LslWriteToSvr

Corresponds to [LslWriteToSvrH\(\)](#) function with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvr(
    uint32_t    objAddr,
    uint32_t    dValue
);

```

6.1.15.20 LslWriteToSvrEx

Corresponds to [LslWriteToSvrExH\(\)](#) with ocbNum = 0.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrEx(
    plcptr_t    objAddr,
    uint32_t    dValue,
    uint32_t    *pResult
);

```

6.1.15.21 LslWriteToSvrExH

This function uses the NewInst method of a server to call its Write() method in order to change the server value.

```

extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrExH(
    int32_t    ocbNum,
    plcptr_t    objAddr,
    uint32_t    dValue,
    uint32_t    *pResult
);

```

Transfer parameters	Type	Description
ocbNum	int32_t	Online connection block number (return value of OcbOpen())
objAddr	plcptr_t	Server channel address
dValue	uint32_t	New value to be written to the server
pResult	uint32_t*	Pointer to a variable where the return value of the NewInst method is entered

Return parameters	Type	Description
		<p>TRUE Function successful</p> <p>FALSE Error</p>
To obtain extended error information, call the GetLastError() function.		

Example

```
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

Calls the Write() method of a server, in order to change the server value.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrH(
    int32_t    ocbNum,
    uint32_t   objAddr,
    uint32_t   dValue
);
```

Transfer parameters	Type	Description
ocbNum	int32_t	Online connection block number (return value of OcbOpen())
objAddr	uint32_t	Server channel address
dValue	uint32_t	New value, which should be written to the server
Return parameters	Type	Description
		<p>TRUE Success</p>

		FALSE	Error
--	--	-------	-------

Example

```
static char* GetChannelTypeStr(ChMode Mode); //Siehe LslGetObjCls()

static void TestWriteToSvr()
{
    uint32_t          ObjAddr;
    ChModeMode;
    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");
            }
        }
        else
        {
            printf("LslGetObject() failed 0x%08X\n", GetLastError());
        }
    }
    else
    {
        printf("Error Online:0x%08X \n", GetLastError());
    }
}
```

```
    ::Offline();
}
```

6.1.15.23 LslWriteToSvrStr

Corresponds to LslWriteToSvrStrH() with ocbNum = 0.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrStr(
    plcptr_t    objAddr,
    const char  *pString,
    int32_t      iLen
);
```

6.1.15.24 LslWriteToSvrStrH

This function calls the NewInst method of a string server to write the string.

```
extern "C" LSL_BOOL LASAL32_EXPORTS LslWriteToSvrStrH(
    int32_t      ocbNum,
    plcptr_t    objAddr,
    const char  *pString,
    int32_t      iLen
);
```

Transfer parameters		Type	Description				
ocbNum		int32_t	Online connection block number (return value of OcbOpen())				
objAddr		plcptr_t	Server channel address				
pString		const char*	Pointer to the 0-terminated ASCII or Unicode string that is to be written to the server. If NULL is passed here, an empty string is written.				
iLen		int32_t	Number of bytes to be written or -1 for the entire string				
Return parameters		Type	Description				
			<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50px; text-align: center;">TRUE</td><td>Fuction successful</td></tr> <tr> <td style="text-align: center;">FALSE</td><td>Error</td></tr> </table> <p>To obtain extended error information, call the GetLastError() function.</p>	TRUE	Fuction successful	FALSE	Error
TRUE	Fuction successful						
FALSE	Error						

Example

```
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 in the Loader

6.2.1 Introduction

Comlink provides the rules for visualization computers (client) and PLCs (server) for exchanging data. The client requests data from the server and the server sends the requested data in response to the client.

The client either sends a request to the server and waits for the answer or it places an update cell into the server's update buffer. An update cell in the update buffer instructs the server to continuously monitor input data in the PLC and send responses to the client when the value of the input data changes.

The server maintains two different update buffers: A static and a dynamic update buffer. The client decides where the update cells are stored. The static update buffer is typically initialized with update cells during the project start-up, after which, it is no longer changed. The content of the dynamic update buffer however, is changed when new variables are required. E.g. when the user selects a screen with new variables.

6.2.2 The Comlink Interface in LASAL Class

The Comlink interface in LASAL Class consists of the following functions:

- Login
- TxCommand
- TxUpd
- StartStopRefresh
- InstallCallback

- Logout
- LDR_SetCanWait
- LDR_SetCanWaitRemote
- LDR_SetRs232ComlinkParams

To communicate with a Comlink server, the following steps should be taken:

- Login to the server with function Login.
- Applications that want to communicate with the command interpreter of the comlink server, e.g. to exchange data of an objects server channel, should call TxCommand.
- Applications that want to be notified of data value changes in an object without having to query it should use the functions TxUpd, StartStopRefresh and InstallCallBack.

The interface functions are implemented in the loader. If the Comlink functions are not needed in the project, memory can be saved by using a loader that does not contain the Comlink code or only parts thereof. To use a customized loader, the following steps should be taken:

- Open the project Loader, located in the subdirectory loader of the LASAL Class program directory.
- Modify the following loader.h definitions:

```
#define COMLINK_LASAL // comment out, when the whole comlink code should be excluded
#define COMLINK_TCP_SERVER // comment out, when you don't need a TCP comlink server
#define COMLINK_TCP_CLIENT // comment out, when you don't need a TCP comlink client
```
- Rebuild the loader project.
- Link the user project.
- It is important to note that these changes are lost when a new Lasal Class is installed.

The function prototypes of the Comlink functions can be found in the include file ComTypes.h.



All interface functions except InstallCallback can block and wait for the answer of the server if the interface type is not LOCAL or INTERN. These functions should not be called in real-time or cyclic tasks, because they could delay other real-time or cyclic work!

6.2.3 Assigning CAN Identifiers

The following ranges of CAN identifiers are reserved for the comlink protocol:

- 16#700 – 16#71F
- 16#500 – 16#67F (provided that the default value for the number of max. comlink channels is not changed)

The CAN object identifiers that are actually allocated depend on the following factors:

- CAN station number of the PLC (STATION), range: 0-31
- Number of outgoing comlink connections (NBR_CONNECTIONS)
- Max. number of comlink channels (MAX_CHANNELS). The default value is 5. This value can be customized by changing the #define COMLINK_CAN_COMCHS in the loader (Lasal1: loader.h, Lasal2: UserDef.h). After changing this value, the loader has to be compiled and linked with the application. Note that all PLCs that communicate over the comlink protocol have to use the same value !

These are the formulas to calculate the allocated 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
```

Example

3 PLCs with station numbers 0, 11, 25.

Station 11 and 25 establish a comlink connection to station 0.

The default value for COMLINK_CAN_COMCHS is untouched (-> 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
```

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

6.2.3.1 InstallCallBack

The InstallCallBack() function installs a callback function for all existing login sessions. The callback function is used when value of an update cell changes.

```
FUNCTION GLOBAL __CDECL InstallCallBack
VAR_INPUT
    pCallback : ^void;
END_VAR;
```

Transfer parameters	Type	Description
pCallback	^VOID	<p>Pointer to a callback function with the following prototype:</p> <pre>FUNCTION GLOBAL __CDECL PrototypeCallback VAR_INPUT pComdef : ^COMDEF; pData : ^d2LSE; END_VAR;</pre>

Callback Fuction

Transfer parameters	Type	Description
PComdef	^COMDEF	Pointer to a COMDEF structure. This pointer can be used to identify the data source.
pData	^d2LSE	<p>Pointer to a d2LSE structure with the following layout:</p> <pre>d2LSE : STRUCT VarlistID : UDINT; uiOffs : UINT; Data : DINT; END_STRUCT;</pre>

Structure d2LSE

VarlistID	UDINT	This is the number specified in the function TxUpd for the Varlist-ID element of the pLsIcommregdata parameter
-----------	-------	----------------------------------------------------------------------------------------------------------------

UiOffs	UINT	0-based index of the updated cell in the update buffer. A value from 0-999 indicates that the update cell comes from the static update buffer and a value greater than 1000 indicates that it comes from the dynamic update cell.
data	DINT	Update cell data

6.2.3.2 LDR_SetCanWait

The LDR_SetCanWait() sets the time delay between two sends of a comlink CAN message. This delay is necessary to prevent overloading the can bus. The default value for the delay is 2 ms.

```
FUNCTION LDR_SetCanWait
VAR_INPUT
    us_wait : usint;
END_VAR
VAR_OUTPUT
    old : usint;
END_VAR;
```

Transfer parameters	Type	Description
us_wait	USINT	Delay time in ms
Return parameters	Type	Description
old	USINT	Previous delay value

6.2.3.3 LDR_SetCanWaitRemote

The LDR_SetCanWaitRemote() sets the time delay between two sends of a comlink CAN message from a remote station. This delay is necessary to prevent overloading the can bus. The default value for the delay is 2 ms.

```
FUNCTION LDR_SetCanWaitRemote
VAR_INPUT
    pComdef : ^COMDEF;
    wait : INT;
END_VAR
VAR_OUTPUT
    retVal : DINT;
END_VAR;
```

Transfer parameters	Type	Description
pComdef	^COMDEF	Pointer to a COMDEF structure

wait	INT	Delay time in ms
Return parameters	Type	Description
RetVal	DINT	0 Success ≠0 Error code

6.2.3.4 LDR_SetRs232ComlinkParams

The LDR_SetRs232ComlinkParams() function can be used to set the parameters for a serial interface for use with a comlink connection.

```
FUNCTION GLOBAL __cdecl LDR_SetRs232ComlinkParams
VAR_INPUT
    interface      : UDINT;
    baudrate      : UINT;
    startServer   : DINT;
    fnInitInterface : pVoid;
    pThis          : pVoid;
END_VAR
VAR_OUTPUT
    retVal        : DINT;
END_VAR;
```

Transfer parameters	Type	Description
interface	UDINT	Interface number for the serial interface. The constants are defined in the Comtypes.h (COMLINK_IFNUM_COM1, COMLINK_IFNUM_COM2, etc.) header file.
baudRate	UINT	The baudrate setting. The constants are defined in the Isl_st_serial.h (SERUSERBAUD_9600, SERUSERBAUD_19600 etc.) header file. A value of 16#FFFF means the default baudrate defined in the loader is used.
startServer	DINT	When this flag is set to 1 then the comlink server opens the serial interface and responds to incoming requests. A value of 0 does not open the interface. For the comlink server to use the serial interface, it must be requested explicitly by calling this function. If the comlink server opens the serial interface without being requested, the existing applications that use the serial interface might not function.
fnInitInterface	pVoid	A pointer to a callback function with the following prototype: <code>FUNCTION GLOBAL __cdecl LDR_InitRs232Interface</code> <code>VAR_INPUT</code> <code> pThis : pVoid;</code> <code> interfaceNbr : UDINT;</code> <code> hComm : pVoid;</code>

		<pre> END_VAR VAR_OUTPUT // 1 : initialized // 0 : init in progress //<0 : init failed initStatus : DINT; END_VAR; </pre> <p>When the value of this parameter points to a callback function, the comlink software calls this function after the serial interface has been initialized and before the communication with other comlink nodes starts. The return value of this function tells the caller when the communication can start. A value of 1 means that communication can start. Once communication has started, this function is called at regular intervals so that the application can stop the communication when a change has occurred and to take control of the serial interface (e.g. to terminate an existing modem connection). This function is normally used to establish a dial-up connection with a modem.</p> <p>The hComm parameter is a handle for the opened serial interface, which must be passed as a parameter to the SERUSER API functions.</p>				
pThis	pVoid	This value is passed as the pThis parameter to the fnInitInterface callback function. It can be used to pass an object pointer to the global callback function.				
Return parameters	Type	Description				
retVal		<table border="1"> <tr> <td>0</td><td>Success</td></tr> <tr> <td>#0</td><td>Error code</td></tr> </table>	0	Success	#0	Error code
0	Success					
#0	Error code					

6.2.3.5 Login

The Login() function tries to establish a connection to the server. If successful, the function creates a comlink resource and initializes a COMDEF structure that can be used to access the resource in other functions.

```

FUNCTION GLOBAL __CDECL Login
VAR_INPUT
    pComdef : ^COMDEF;
END_VAR
VAR_OUTPUT
    result : UINT;
END_VAR;

```

Transfer parameters	Type	Description
pComdef	^COMDEF	Pointer to a COMDEF structure

Return parameters	Type	Description	
result	UINT	0	Success

#0 [Error code](#)

A return value of 16#FFE (connection not established) could mean that the server is not up. In this case, the Login() function can be retried later. Use the Logout() function to clear the connection.

Structure COMDEF

```
Comdef : STRUCT
  Interface : UDINT;
  Address   : UDINT;
  pt_COM    : ^COMDATA;
  IPAdresse : UDINT;
  port      : UDINT;
  res       : UDINT;
END_STRUCT;
```

The Interface, Address and IPAdresse elements must be specified by the user.

Interface	UDINT	Specifies the interface type and can be one of the following values: <ul style="list-style-type: none"> 0 LOCAL: The server runs on the same machine as the client. 1 INTERN: Same as LOCAL 6 CAN1: First CAN interface 8 TCP/IP1: First TCP/IP interface. This value is obsolete; it is replaced by value 10 and available for compatibility reasons only. 10 TCP/IP1: First TCP/IP interface
Address	UINT	Specifies the server's CAN station number. The available station numbers range from 0 to 31. This parameter is only valid, when a CAN interface type is used.
pt_COM	^COMDATA	The Login-function stores a pointer to a communication structure for this comlink resource in element pt_COM.
IPAdresse	UDINT	Specifies the IP-address of the server. The IP-address is stored in a UDINT where the hi-byte contains the first number of the address. This parameter is only valid, when a TCP/IP interface is used.
port	UDINT	Specifies the server's CAN station number. This parameter is only valid, when a TCP/IP interface with a value ≥ 10 is used. Port

		number 0 means that the default value should be used.
res	UDINT	Reserved element and must be set to zero

6.2.3.6 StartStopRefresh

The StartStopRefresh() function tells the server how many update cell entries in an update buffer should be scanned. If the value of an update cell changes, the callback function informs the client of the new value.

```
FUNCTION GLOBAL __CDECL StartStopRefresh
VAR_INPUT
  pComdef : ^COMDEF;
  count   : UINT;
  typ     : UINT;
END_VAR;
```

Transfer parameters	Type	Description				
PComdef	^COMDEF	Pointer to a COMDEF structure that was initialized by Login()				
count	UINT	Specifies the number of update cells that should be scanned				
typ	UINT	Specifies the type of the update buffer <table border="1" style="margin-left: 20px;"> <tr> <td>0</td> <td>Static update buffer</td> </tr> <tr> <td>1</td> <td>Dynamic update buffer</td> </tr> </table>	0	Static update buffer	1	Dynamic update buffer
0	Static update buffer					
1	Dynamic update buffer					

6.2.3.7 TxCommand

The TxCommand() function sends a command to the interpreter and waits for the answer.

```
FUNCTION GLOBAL __CDECL TXCOMMAND
VAR_INPUT
  Command  : UDINT;
  length   : UDINT;
  charptr  : ^USINT;
  pComdef  : ^COMDEF;
  Presu    : ^UDINT;
END_VAR
VAR_OUTPUT
  Status   : IPRSTATES;
END_VAR;
```

Transfer parameters	Type	Description
Command	UDINT	Command to the interpreter

length	UDINT	Length of the data in charptr
charptr	^USINT	Pointer to the parameters of the interpreter command
pComdef	^COMDEF	Pointer to a COMDEF structure that was initialized during login
Presu	^UDINT	Pointer to an UDINT that receives the address of the result buffer of the interpreter
Return parameters	Type	Description
Status	IPRSTATES	This function returns the state of the interpreter, e.g. READY, ERROR, BUSY etc. When no error occurs READY (0) is returned. A return value of ERROR (1) is not necessarily an error in the interpreter of the server. It can also indicate a missing connection or any other error condition in the client. To distinguish between an interpreter error and any other error, use function TxCommandEx() .

This function is not thread safe or sequence invariant for a single channel. This means that a TxCommand() cannot be called from two threads for the same channel simultaneously. In addition, a TxCommand() call could end in an infinite loop when a read or a write method is called via TxCommand(command I_READ or I_WRITE) that contains another TxCommand() call.

It is possible however, to call TxCommand() for one channel in one thread and for another channel on a different thread.

When thread-safety is required, use the [TxCommandEx\(\)](#) function.

6.2.3.8 TxCommandEx

The TxCommandEx() function sends a command to the interpreter and waits for the answer. Compared to the [TxCommand\(\)](#) function, this function is the thread-safe version (starting with 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;
```

Transfer parameters	Type	Description

Command	UDINT	Command to the interpreter
length	UDINT	Length of the data in charptr
charptr	^USINT	Pointer to the parameters of the interpreter command
pComdef	^COMDEF	Pointer to a COMDEF structure that was initialized during login
pResubuf	^RESULTS	Pointer to a buffer that receives the result of the interpreter
sizeOfResuBuf	UDINT	Size of the result buffer
pReason	^comlinkReason	Pointer to a variable that receives the reason-code in the case of an error; this pointer can be NIL
Return parameters	Type	Description
Status	IPRSTATES	This function returns the state of the interpreter, e.g. READY, ERROR, BUSY etc. When no error occurs READY (0) is returned. A return value of ERROR (1) is not necessarily an error in the interpreter of the server. It can also indicate a missing connection or any other error condition in the client. To distinguish between an interpreter error and any other error, use parameter pReason to find error source. The values for the error codes are defined in the header file ComTypes.h (e.g. COMLINK_ERR_NOCONNECTION).

This function is thread safe for a single channel. This means that a TxCommand can be called from two threads for the same channel simultaneously. When TxCommandEx is called while it's being run by another thread, the second call of the function is blocked until the first call is finished.

6.2.3.9 TxUpd

The TxUpd function transmits an update cell to the update buffer.

```
FUNCTION GLOBAL __cdecl TxUpd
VAR_INPUT
    pLslcommregdata : ^Lslcommregdata;
    pComdef        : ^COMDEF;
END_VAR;
```

Transfer parameters	Type	Description
pLslcommregdata	^Lslcommregdata	pointer to a structure that defines the properties of the update cell
PComdef	^COMDEF	pointer to a COMDEF structure that was initialized by Login

Structure Lslcommregdata

```
LslCommRegData : STRUCT
    LASALID      : UDINT;
```

```

Channel      : UINT;
VarPos       : UINT;
uiTIME       : UINT;
VarlistID    : UDINT;

```

END_STRUCT;

LASALID	UDINT	Specifies the Lasal ID of the object. The Lasal ID can be obtained with the interpreter command I_GET_OBJ.
Channel	UINT	This function function does not use the Channel element.
VarPos	UINT	Specifies a zero-based index of the update cell in the update list.
uiTime		Specifies the update rate for the update cell. Bits 0-14 specify the update rate, in milliseconds. When the highest order Bit (Bit 15) is set, the comlink reports the CRC for a VirtualBaseInit object. Otherwise, it reports the result of the read-method from the server channel. When a CRC is reported, the content of a data buffer has changed.
VarlistID	UDINT	Specifies a number that is not interpreted or affected by the comlink application. Varlist ID can be used to associate a value reported by the server with an entry in the client's database.

6.2.4 Error Codes

The following table describes the error codes returned from comlink client functions.

Code	Description
16#FFF0	A general unrecoverable error has occurred and provides no information on the cause.
16#FFF9	The server sent an invalid response.
16#FFFA	The specified address is already in use.
16#FFFFB	The client interface functions are not available due to an unrecoverable error during initialization.
16#FFFC	The specified interface type is not supported.
16#FFFD	Maximum number of connections has been exceeded.
16#FFFE	A connection to the server could not be established or an existing connection was removed. This may be a temporary error; therefore the function that triggered this error can be recalled.
16#FFFF	An invalid parameter has been specified.

6.3 Library for SIGMATEK Hardware Access

6.3.1 General

With the SIGMATEK library StkLib, the SIGMATEK Hardware (e.g. an IPC) can be accessed with Microsoft system software applications. The programming interface is implemented as a DLL (Dynamic Link Library). Windows drivers are required depending on the type of the hardware.

For the following hardware, StkLib API functions are available:

- CAN bus adaptor BU104
- CAN hardware of an IPC
- DIAS IPC hardware (standard DIAS master)

6.3.2 Installation

With the DrvSetup.exe program, the DLLs and StkLib drivers can be installed. The installation program provides a choice of software components to install.

Sample programs for Microsoft Visual C++ are also installed in the LASAL program directory (usually under C:\Programs\Lasal) under the subdirectory online\driver\visualC6\StkLib.

6.3.3 Requirements

Microsoft Windows 98, Windows ME, Windows 2000 or Microsoft Windows XP.

6.3.4 CAN APIs

With the CAN API functions, the CAN controller for a BU104 CAN bus interface or the CAN hardware of an IPC can be accessed. An Intel 82526 is used in the BU104 and an Intel 82527 CAN controller in the IPC. The functional range of the Intel 82526 is lower than that of the Intel 82527. The description of the API functions makes reference to certain functions that are not available for the BU104.

Number of available communication objects:

BU104	The number of the possible communication objects depends on the size of the individual objects. A total of 56 bytes are available for communication objects. For one object, 3 bytes are needed in addition to the length of the data in the CAN object.
IPC	15 communication objects (one of which is a basic CAN object).

6.3.4.1 StkCanAddObject

Adds a CAN object to the CAN controller.

```
extern "C" int StkCanAddObject(
    HANDLE         handle,
    unsigned int   identifier,
    unsigned int   dataLengthCode,
    unsigned int   flags,
    unsigned int   localMask,
    unsigned int   rxBufQueueSize,
    BYTE          *initialData
);
```

Transfer parameters	Type	Description									
handle	HANDLE	CAN driver handle									
identifier	UINT	Message identifier									
dataLengthCode	UINT	Length of the message object. The valid length values range from 0 to 8.									
flags	UINT	<p>Attributes of the message object. It is possible to specify a combination of the following values.</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>DIR_TX</td> <td>0</td> <td>Send; without this attribute: receive</td> </tr> <tr> <td>XTD_IDEN TIFIER</td> <td>1</td> <td> <p>The message object uses a 29-bit extended identifier. Without this attribute, a standard 11-bit identifier is used.</p> <p>In the BU104, no extended identifiers can be used.</p> </td> </tr> </tbody> </table>	Name	Value	Meaning	DIR_TX	0	Send; without this attribute: receive	XTD_IDEN TIFIER	1	<p>The message object uses a 29-bit extended identifier. Without this attribute, a standard 11-bit identifier is used.</p> <p>In the BU104, no extended identifiers can be used.</p>
Name	Value	Meaning									
DIR_TX	0	Send; without this attribute: receive									
XTD_IDEN TIFIER	1	<p>The message object uses a 29-bit extended identifier. Without this attribute, a standard 11-bit identifier is used.</p> <p>In the BU104, no extended identifiers can be used.</p>									
localMask	UINT	This parameter is the local mask for the 'Basic CAN Object'. "0" means "don't care". "1" means "must match". This means that the identifier of the receiving message must match the configured message identifier with the appropriate bit position. The 'Global Mask' does not exist in the BU104.									
rxBufQueueSize	UINT	Receiving messages are put into a queue in the driver. This parameter shows the number of CAN messages that can be buffered in the queue of the message object.									
initialData	BYTE	Pointer to an array, which contains data bytes that are written to the message object. If remote frames are used, the data for the send object must be initialized. The value ZERO indicates that the message object data is initialized with 0.									
Return parameters	Type	Description									

		<p>>0 Positive number that can be used to identify the message object in the CAN controller</p> <p><0 Negative error code</p>
--	--	-------------------------------------------------------------------------------------------------------------------------------------

6.3.4.2 StkCanClose

Closes a CAN driver handle

```
extern "C" void StkCanClose(HANDLE handle);
```

Transfer parameters	Type	Description
handle	HANDLE	CAN driver handle

6.3.4.3 StkCanOpen

Opens a CAN driver and returns a handle, which is transferred as parameter for additional CAN API functions.

```
extern "C" HANDLE StkCanOpen(
    char           *driverName,
    unsigned int   flags
);
```

Transfer parameters	Type	Description								
driverName	CHAR	<p>Driver name</p> <table> <tr> <td>Name</td> <td>Meaning</td> </tr> <tr> <td>CANLPT1</td> <td>CAN bus adapter BU104 on the LPT1 connection</td> </tr> <tr> <td>CANLPT2</td> <td>CAN bus adapter BU104 on the LPT2 connection</td> </tr> <tr> <td>CANIPC1</td> <td>CAN hardware of the IPC</td> </tr> </table>	Name	Meaning	CANLPT1	CAN bus adapter BU104 on the LPT1 connection	CANLPT2	CAN bus adapter BU104 on the LPT2 connection	CANIPC1	CAN hardware of the IPC
Name	Meaning									
CANLPT1	CAN bus adapter BU104 on the LPT1 connection									
CANLPT2	CAN bus adapter BU104 on the LPT2 connection									
CANIPC1	CAN hardware of the IPC									
flags	UINT	Reserved; must to be set to 0								
Return parameters	Type	Description								
		If no error occurs, the open handle is returned to the CAN driver. Otherwise, INVALID_HANDLE_VALUE is returned.								

6.3.4.4 StkCanRxObjectAny

This function receives a CAN message from an arbitrary message object in the 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
);
```

Transfer parameters		Type	Description						
handle		HANDLE	CAN driver handle						
pIdentifier		UINT	Pointer to a variable in which the message identifier of the receiving message is saved						
pFlags		UINT	Pointer to a variable in which the attributes of the received message are saved						
pData		BYTE	Pointer to an array in which the data of received message is saved						
sizeOfData		UINT	Size of the pData array in bytes						
pdataLengthCode		UINT	Pointer to a variable in which the length of the receiving message is saved. It is important to remember that the BU104 CAN controller always reports the configured length and not the length of the received message.						
timeout100ns		UNIT	Timeout in 100 ns units; 0 indicates that a standard value should be used						
Return parameters		Type	Description						
			<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;"></td><td style="width: 10%;">0</td><td>OK</td></tr> <tr> <td></td><td><0</td><td>Negative error code</td></tr> </table>		0	OK		<0	Negative error code
	0	OK							
	<0	Negative error code							

6.3.4.5 StkCanRxObject

This function receives a CAN message from a corresponding message object in the 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
);

```

Transfer parameters	Type	Description
handle	HANDLE	CAN driver handle
msgObjNr	UINT	Number of the message object in the CAN controller (return value of StkCanAddObject())
pIdentifier	UINT	Pointer to a variable in which the message identifier of the receiving message is saved
pData	BYTE	Pointer to an array in which the receiving message data is saved
sizeOfData	UINT	Size of the pData array in bytes
pDataLengthCode	UINT	Pointer to a variable in which the length of the receiving message is saved. It is important to remember that the BU104 CAN controller always reports the configured length and not the length of the received message.
timeout100ns	UINT	Timeout in 100 ns units; 0 indicates that a standard value should be used
Return parameters	Type	Description
		0 OK <0 Negative error code

6.3.4.6 StkCanSetup

Positioning of the CAN controller's configuration parameters.

```

extern "C" int StkCanSetup(
    HANDLE      handle,
    unsigned int baudRate,
    unsigned int globalMaskStd,
    unsigned int globalMaskExt
);

```

Transfer parameters	Type	Description
handle	HANDLE	CAN driver handle
baudRate	UINT	CAN baud rate

		<table border="1"> <tr><td>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	'Global Mask' for CAN messages with a 'standard CAN Identifier'																
globalMaskExt	UINT	'Global Mask' for CAN messages with an extended CAN Identifier'																
Return parameters	Type	Description																
		<table border="1"> <tr><td>0</td><td>OK</td></tr> <tr><td><0</td><td>Negative error code</td></tr> </table>	0	OK	<0	Negative error code												
0	OK																	
<0	Negative error code																	

The "Global Mask" is used for 'message acceptance filtering'. For this reason, the bits in the CAN identifier of a receiving message can be masked with "don't care". "0" means "don't care". "1" means "must match". This means that the identifier of the receiving message must match the configured message identifier with the appropriate bit position. The 'Global Mask' does not exist in the BU104.

6.3.4.7 StkCanSwReset

Triggers a software reset in the CAN controller.

```
extern "C" int StkCanSwReset(HANDLE handle);
```

Transfer parameters	Type	Description				
handle	HANDLE	CAN driver handle				
Return parameters	Type	Description				
		<table border="1"> <tr><td>0</td><td>OK</td></tr> <tr><td><0</td><td>Negative error code</td></tr> </table>	0	OK	<0	Negative error code
0	OK					
<0	Negative error code					

If the CAN controller is in the BUS-OFF state, it must be reset through the software. The software reset for the CAN hardware in the IPC does not have to be executed by the application program. In this case, the driver executes the software reset automatically. A return value of, 1004/CANERR_DRVR_BUS_STATUS, means that the CAN controller status is in the BUS-OFF state.

6.3.4.8 StkCanTxObject

Sends a CAN message.

```
extern "C" int StkCanTxObject(
    HANDLE      handle,
    unsigned int msgObjNbr,
    unsigned int identifier,
    unsigned int dataLengthCode,
    unsigned int flags,
    BYTE        *pData,
    unsigned int timeout100ns
);
```

Transfer parameters	Type	Description								
handle	HANDLE	CAN driver handle								
msgObjNr	UINT	Number of the message object in the CAN controller (return value of StkCanAddObject())								
identifier	UINT	Message identifier								
dataLengthCode	UINT	Length of the message; the valid length values range from 0 to 8								
flags	UINT	Attributes of the message object <table border="1" style="margin-top: 10px; width: 100%;"> <tr> <th>Name</th> <th>Value</th> <th>Meaning</th> </tr> <tr> <td>XTD_IDENTIFIER</td> <td>1</td> <td> The message object uses a 29-bit extended identifier. Without this attribute, a standard 11-bit identifier is used. In the BU104, no extended identifiers can be used. </td> </tr> </table>			Name	Value	Meaning	XTD_IDENTIFIER	1	The message object uses a 29-bit extended identifier. Without this attribute, a standard 11-bit identifier is used. In the BU104, no extended identifiers can be used.
Name	Value	Meaning								
XTD_IDENTIFIER	1	The message object uses a 29-bit extended identifier. Without this attribute, a standard 11-bit identifier is used. In the BU104, no extended identifiers can be used.								
pData	BYTE	Pointer to an array containing the data to be sent								
Timeout100ns	UINT	Timeout in 100 ns units; 0 indicates that a standard value should be used								
Return parameters	Type	Description								
		0 OK								

		<0 Negative error code
--	--	------------------------

6.3.5 DIAS APIs

The DIAS API functions allow access to the DIAS hardware (standard DIAS Master) in an IPC. It is not possible to react to DIAS Master interrupts.

Up to 64 DIAS modules can be connected to the DIAS bus. The DIAS module is assigned an address within a value range from 0-63. The DIAS module has an address range of 256 data bytes and 256 control register bytes. The meaning of these bytes depends on the DIAS module type. For example, the outputs 0-15 of a DTO163 digital output module can be addressed by writing the word to the data address 0. The module identifier for a DIAS module can be found in the control-register-byte at address 255. Further information regarding the applicable DIAS module types can be found in the DIAS documentation.

The module-independent control registers of the DIAS master are located in a 16-kbyte address range.

6.3.5.1 StkDiasCheckError

Provides the content of the DIAS error register.

```
extern "C" int StkDiasCheckError(
    HANDLE handle
);
```

Transfer parameters	Type	Description
handle	HANDLE	CAN driver handle
Return parameters	Type	Description
		0 OK <0 Negative error code

6.3.5.2 StkDiasClose

Closes a CAN driver handle.

```
extern "C" void StkDiasClose(HANDLE handle);
```

Transfer parameters	Type	Description

handle	HANDLE	CAN driver handle
--------	--------	-------------------

6.3.5.3 StkDiasGetModID

Supplies the DIAS ID of a module for a specified address.

```
extern "C" int StkDiasGetModID(
    HANDLE handle,
    BYTE mod,
    BYTE *id
);
```

Transfer parameters	Type	Description
handle	HANDLE	CAN driver handle
mod	BYTE	Module address (0-63)
id	BYTE	Pointer to a variable in which the DIAS ID is saved. If there is no module is located at this address, the value 255 is returned.
Return parameters	Type	Description
		0 OK <0 Negative error code

6.3.5.4 StkDiasOpen

Opens a DIAS driver and returns a handle as a parameter for further DIAS API functions.

```
extern "C" HANDLE StkDiasOpen(
    char *driverName,
    unsigned int flags
);
```

Transfer parameters	Type	Description
driverName	CHAR	Driver name Name Meaning CANIPC1 DIAS IPC hardware (standard DIAS master)
flags	UINT	Reserved; must be set to 0
Return parameters	Type	Description

		0 OK
		<0 Negative error code

6.3.5.5 StkDiasReadByte

Reads a byte from the data area of a DIAS module.

```
extern "C" int StkDiasReadByte(
    HANDLE handle,
    BYTE mod,
    WORD ofs,
    BYTE *pData
);
```

Transfer parameters		Type	Description
handle		HANDLE	CAN driver handle
mod		BYTE	Module address (0-63)
ofs		WORD	Offset in the module address area
pData		BYTE	Pointer to a variable in which the read byte is saved
Return parameters		Type	Description
			0 OK
			<0 Negative error code

6.3.5.6 StkDiasReadCtrl

Reads the control register byte of a DIAS module.

```
extern "C" int StkDiasReadCtrl(
    HANDLE handle,
    BYTE mod,
    WORD ofs,
    BYTE *pData
);
```

Transfer parameters		Type	Description
handle		HANDLE	CAN driver handle
mod		BYTE	Module address (0-63)

ofs	WORD	Offset in the module address area
pData	BYTE	Pointer to a variable in which the read byte is saved
Return parameters	Type	Description
		0 OK <0 Negative error code

6.3.5.7 StkDiasReadWord

Reads a word out of the data range of an adequate DIAS module.

```
extern "C" int StkDiasReadWord(
    HANDLE handle,
    BYTE mod,
    WORD ofs,
    WORD *pData
);
```

Transfer parameters	Type	Description
handle	HANDLE	CAN driver handle
mod	BYTE	Module address (0-63)
ofs	WORD	Offset in the module address area
pData	WORD	Pointer to a variable in which the read word is saved
Return parameters	Type	Description
		0 OK <0 Negative error code

6.3.5.8 StkDiasReadRegister

Reads the control register byte from the DIAS address area.

```
extern "C" int StkDiasReadRegister(
    HANDLE handle,
    WORD ofs,
    BYTE *pData
);
```

Transfer parameters	Type	Description

handle	HANDLE	CAN driver handle
ofs	WORD	Register offset in the DIAS address area
pData	BYTE*	Pointer to a variable in which the read byte is saved
Return parameters	Type	Description
		0 OK <0 Negative error code

6.3.5.9 StkDiasWriteByte

Writes a byte into the data range of an adequate DIAS module.

```
extern "C" int StkDiasWriteByte(
  HANDLE handle,
  BYTE mod,
  WORD ofs,
  BYTE data
);
```

Transfer parameters		Type	Description
handle	HANDLE	CAN driver handle	
mod	BYTE	Module address (0-63)	
ofs	WORD	Offset in the module address area	
data	BYTE	Byte to be written	
Return parameters	Type	Description	
		0 OK <0 Negative error code	

6.3.5.10 StkDiasWriteCtrl

Reads the control register byte of a DIAS module.

```
extern "C" int StkDiasWriteCtrl(
  HANDLE handle,
  BYTE mod,
  WORD ofs,
  BYTE data
);
```

Transfer parameters		Type	Description	
handle		HANDLE	CAN driver handle	
mod		BYTE	Module address (0-63)	
ofs		WORD	Offset in the module address area	
data		BYTE	Byte to be written	
Return parameters		Type	Description	
			0	OK
			<0	Negative error code

6.3.5.11 StkDiasWriteRegister

Reads the control register byte from the DIAS address area.

```
extern "C" int StkDiasWriteRegister(
    HANDLE handle,
    WORD    ofs,
    BYTE    data
);
```

Transfer parameters		Type	Description	
handle		HANDLE	CAN driver handle	
ofs		WORD	Register offset in the DIAS address area	
data		BYTE	Byte to be written	
Return parameters		Type	Description	
			0	OK
			<0	Negative error code

6.3.5.12 StkDiasWriteWord

Writes a word to the data area of a DIAS module.

```
extern "C" int StkDiasWriteWord(
    HANDLE handle,
    BYTE    mod,
    WORD    ofs,
    WORD    wData
```

);

Transfer parameters		Type	Description
handle		HANDLE	CAN driver handle
mod		BYTE	Module address (0-63)
ofs		WORD	Offset in the module address area
wData		WORD	Word to be written
Return parameters		Type	Description
			0 OK
			<0 Negative error code

6.4 API Safety DLL

6.4.1 General Information

This document contains a description of the 'Safety DLL Interface' functions. These functions are used to create a connection for communication from a standard PLC to Safe CPU and perform the following procedures:

- Download a configuration
- Set the Verified flags
- Change passwords
- Start the safe CPU (exit the service mode)
- Stop the safe CPU (restore the service mode)
- Query status
- Restart the Safety application
- Cancel an error



The functions cannot be used for running a process fully automatically. The program provider must ensure that the input data is from the user and not hard coded into the program.

If functions that should change the file or connection state fail, the file and connection state remain unchanged.

6.4.2 Interface Function ISAFETY_DLL

The functions are then available in the LasalOS after the Safety DLL is loaded. The safety.dlm must be thereby located in the C:\LSLSYS directory and the CLI command loadsafety run. With the Salamander operating system, this step is not required.

To use the interface, the lsl_st_safety.dll.h header file must be linked and a pointer to the interface structure retrieved via OS_CILGET. This pointer is defined as the first parameter in the macros of the interface functions.

The name of the interface is ISAFETY_DLL and is defined with the INTERFACE_SAFETY_DLL macro.

Example

```
#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
```

The functions are not thread-safe, which means that they cannot be called from several different threads.

Some functions are blocked, which means a certain length to time can pass until the function can be run. These functions therefore, should not be called in a cyclic function of a Lasal object (background, cywork, rtwork) since the processing of the remaining cycles is thereby stopped.

Instead, the functions should be called in their own thread (see MultiTask Interface).

Example

```
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

Changes the password in the target safe CPU.

Before calling this function, [SetUserPromptTime\(\)](#) must be called and a confirmation of the successful transfer requested from the user. After calling [SetUserPromptTime\(\)](#), this function must then be called within 1 to 60 seconds.

```
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(p
```

Transfer parameters		Type	Description	
level	USINT	Login level	1	Debug level
			2	Configuration level
oldPassword	^CHAR	The old password. Size: 8 bytes. If the password is less than 8 characters, the rest must be filled with spaces.		
newPassword	^CHAR	The new password. Size: 8 bytes. If the password is less than 8 characters, the rest must be filled with spaces.	Return parameters	
retval	DINT	0	OK	
		#0	Error code	

Requirements

Connection-State \geq CS_CONNECTION und File-State \geq FS_FILE.

6.4.2.2 SAFETY_CHECK_DONGLE_FW

Checks whether a dongle for the Safety number is plugged in.

Sets the file state to FW_DONGLE, if the module is in dongle mode.

```
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)
```

Transfer parameters	Type	Description	
safetyNbr	UDINT	Safety number of the module with dongle	
Return parameters	Type	Description	
retval	DINT	0	OK
		#0	Error code

Requirements

File-State == FW_FILE

6.4.2.3 SAFETY_CLEAR_CONFIG

Deletes the configuration in the connected module.

```
FUNCTION __CDECL GLOBAL P_Safety_ClearConfig
VAR_OUTPUT
    retval : DINT;
END_VAR;
#define SAFETY_CLEAR_CONFIG(pCil)    pCil^.ClearConfig $ P_Safety_ClearConfig()
```

Return parameters	Type	Description	
retval	DINT	0	OK
		#0	Error code

Requirements

Connection-State == CS_CONNECTION

6.4.2.4 SAFETY_DELETE_STATE

Removes the safety state and must be called at the end. A proper communication connection to the safe CPU has been made.

```
FUNCTION __CDECL GLOBAL P_Safety_DeleteState
VAR_OUTPUT
    retval : DINT;
END_VAR;
#define SAFETY_DELETE_STATE(pCil) pCil^.DeleteState $ P_Safety_DeleteState()
```

Return parameters	Type	Description
		0 OK
		#0 Error code

6.4.2.5 SAFETY_DOWNLOAD_FILE

transmits a part of the previously specified download file to the safe CPU. Before the first call, the number of bytes already transmitted (pBytesTransferred) must be set to 0. The transfer is ended as soon as the number of bytes to transmit equals (pTransferSize).

```
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)
```

Transfer parameters	Type	Description
pBytesTransferred	^UDINT	Pointer to the variable in which the transferred bytes are stored. The value of these variables must be set to 0 before the first call. After calling this function, the current number of bytes that have been transferred is stored in these variables.
pTransferSize	^UDINT	Pointer to the variable in which the total number of bytes to write is stored.

Return parameters	Type	Description	
retval	DINT	0	OK
		#0	Error code

Requirements

Connection-State == CS_LOGGED_IN und File-State == FS_FILE.

6.4.2.6 SAFETY_DOWNLOAD_FW

Transfers a part of a previously defined image file to the safe CPU. Before the first call, the number of bytes already transmitted (pBytesTransferred) must be set to 0. The transfer is ended as soon as the number of bytes to transmit equals (pTransferSize).

```
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)
```

Transfer parameters	Type	Description	
pBytesTransferred		Pointer to the variable in which the transferred bytes are stored. The value of these variables must be set to 0 before the first call. After calling this function, the current number of bytes that have been transferred is stored in these variables.	
pTransferSize		Pointer to the variable in which the total number of bytes to write is stored.	
Return parameters	Type	Description	
retval	DINT	0 OK	
		#0	Error code

Requirements

Connection-State == CS_LOGGED_IN und File-State == FW_DONGLE

6.4.2.7 SAFETY_FILE_GET_PRJNAME

Provides the project name stored in the download file.

```
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(p
```

Transfer parameters		Type	Description	
prjName		^CHAR	Buffer to which the project name is written as a 0-terminated string	
bufSize_prjName		UDINT	Buffer size	
Return parameters		Type	Description	
retval		DINT	0 OK #0 Error code	

Requirements

File-State ≥ FS_FILE

6.4.2.8 SAFETY_FILE_GET_REVNBR

Provides the revision number stored in the download file.

```
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,
```

Transfer parameters		Type	Description	
revNbr		^CHAR	Buffer to which the revision number is written as a 0-terminated string	
bufSize_revNbr		UDINT	Buffer size	

Return parameters	Type	Description	
retval	DINT	0	OK
		≠0	Error code

Requirements

File-State \geq FS_FILE

6.4.2.9 SAFETY_FILE_GET_SCPUNAME

Provides the SCPU name stored in the download file.

```
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
```

Transfer parameters	Type	Description	
scpuName	^CHAR	Buffer to which the SCPU name is written as a 0-terminated string	
bufSize_scpuName	UDINT	Buffer size	
Return parameters	Type	Description	
retval	DINT	0	OK
		≠0	Error code

Requirements

File-State \geq FS_FILE.

6.4.2.10 SAFETY_GET_CFGSTATE

Returns the Configuration status of the 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)
```

Transfer parameters	Type	Description														
pCfgState	^USINT	<p>Pointer to a variable, into which the configuration status is written.</p> <table> <tr><td>1</td><td>INVALID</td></tr> <tr><td>2</td><td>NOT_CONFIGURED</td></tr> <tr><td>4</td><td>CONFIGURED_NOT_DEPLOYED_NOT_VERIFIED</td></tr> <tr><td>8</td><td>CONFIGURED_AND_VERIFIED</td></tr> <tr><td>16</td><td>CONFIGURED_DEPLOYED_NOT_VERIFIED</td></tr> <tr><td>36</td><td>CONFIGURED_NOT_DEPLOYED_NOT_VERIFIED_DEV</td></tr> <tr><td>48</td><td>CONFIGURED_DEPLOYED_NOT_VERIFIED_DEV</td></tr> </table>	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
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															
Return parameters	Type	Description														
retval	DINT	<table> <tr><td>0</td><td>OK</td></tr> <tr><td>#0</td><td>Error code</td></tr> </table>	0	OK	#0	Error code										
0	OK															
#0	Error code															

Requirements

Connection-State ≥ CS_CONNECTION.

6.4.2.11 SAFETY_GET_IMAGE_MOD_ID_FW

Returns the module ID saved in the image file.

```
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

```

Transfer parameters		Type	Description	
pModID		^UDINT	Pointer to the buffer for the module ID (4 bytes)	
Return parameters		Type	Description	
retval		DINT	0	OK
			≠0	Error code

Requirements

File-State == FW_FILE

6.4.2.12 SAFETY_GET_IMAGE_VERSION_FW

Returns the (minor) version saved in the image file.

```

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

```

Transfer parameters		Type	Description	
pVersion		^UDINT	Pointer to the buffer for the version number (4 bytes)	
Return parameters		Type	Description	
retval		DINT	0	OK
			≠0	Error code

Requirements

File-State == FW_FILE

6.4.2.13 SAFETY_GET_MODUL_VERSION_FW

Returns the (minor) version of the module, for which the safety number was given.

```
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(pCil,p1)
```

Transfer parameters		Type	Description
pVersion	^SAFETY_MODUL_VERSION		Pointer to the buffer for the version number (structure of the type SAFETY_MODUL_VERSION)
Return parameters		Type	Description
retval	DINT		0 OK #0 Error code

Requirements

Connection-State == CS_CONNECTION

6.4.2.14 SAFETY_GET_RUNSTATE

Returns the Run status of the 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)
```

Transfer parameters		Type	Description
pRunState	^USINT		Pointer to a variable, into which the Run status is written. 1 POST 2 SERVICE

			3	Error
			8	IDLE
			16	CHK_CFG
			32	OP_TEMP
			64	OP
Return parameters	Type	Description		
retval	DINT	0	OK	
		#0	Error code	

Requirements

Connection-State \geq CS_CONNECTION.

6.4.2.15 SAFETY_GET_SAFETY_NBR

Creates a communication connection to the file stored in the safe CPU and requests the safety number.

```
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)
```

Transfer parameters	Type	Description	
pSafetyNbr	^UDINT	Pointer to the variable into which the requested safety number is written	
Return parameters	Type	Description	
retval	DINT	0	OK
		#0	Error code

Requirements

File-State ≥ FS_FILE.

6.4.2.16 SAFETY_LEAVE_SERVICE_MODE

Instructs the safe CPU to exit the run status SERVICE.

```
FUNCTION __CDECL GLOBAL P_Safety_LeaveServiceMode
VAR_OUTPUT
    retval : DINT;
END_VAR;
#define SAFETY_LEAVE_SERVICE_MODE(pCil)           pCil^.LeaveServiceMode $ P_Safety_LeaveServiceMode
```

Return parameters	Type	Description
retval	DINT	0 OK
		#0 Error code

Requirements

Connection-State == CS_LOGGED_IN.

6.4.2.17 SAFETY_LOGIN

Log in or out of the target Safe CPU.

Before calling this function, [SetUserPromptTime\(\)](#) must be called and the password requested from the user. After calling [SetUserPromptTime\(\)](#), this function must then be called within 1 to 60 seconds.

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)
```

Transfer parameters	Type	Description

level	USINT	Login level
		0 Logout
		1 Debug-Level
Return parameters		
retval	DINT	0 OK
		#0 Error code

Requirement

Connection-State == CS_CONNECTION.

6.4.2.18 SAFETY_NEW_STATE

Creates a new Safety state, which is required for calling the subsequent functions. Must be called at the start. The Safety state remains unchanged until it is removed through SAFETY_DELETE_STATE.

The following elements are stored in the Safety state.

File State	Status of the download files
	FS_IDLE when no download file was entered
	FS_FILE when a download file was entered and could be read
	FS_DOWNLOAD if a download file for the safe CPU was loaded
Connection State	status of the connection to the safe CPU
	CS_IDLE if no safety number was entered
	CS_SAFETY_NBR if a safety number was entered
	CS_CONNECTION if a connection was made
	CS_LOGGED_IN if a login was performed (developer login)
User Prompt Time	Time the SetUserPromptTime was called
Project Name	Name of the project defined in the download file

Revisin Number	Revision number of the project in the download file
S-CPU Name	Name of the destination safe CPU
HW Path	Hardware path for the destination safe CPU

Sets the file state to FS_IDLE and the connection state to CS_IDLE.

```
FUNCTION __CDECL GLOBAL P_Safety_NewState
VAR_OUTPUT
    retval : DINT;
END_VAR;
#define SAFETY_NEW_STATE(pCil) pCil^.NewState $ P_Safety_NewState()
```

Return parameters	Type	Description
retval	DINT	0 OK #0 Error code

6.4.2.19 SAFETY_OPEN_CONNECTION

Creates a communication connection to the target Safe CPU with the previously entered security number. If a download file was specified, the hardware path from the download file is used for the non-Safe network address. Otherwise, all connected Safe CPUs are listed and the one with the appropriate security number is selected.

It is important not that with a stable communication connection, there must be regular communication. Otherwise, it is terminated by the Safe CPU after a timeout of 10 seconds.

Sets the Connection State to CONNECTION

```
FUNCTION __CDECL GLOBAL P_Safety_OpenConnection
VAR_OUTPUT
    retval : DINT;
END_VAR;
#define SAFETY_OPEN_CONNECTION(pCil) pCil^.OpenConnection $ P_Safety_OpenConnection()
```

Return parameters	Type	Description
retval	DINT	0 OK #0 Error code

Requirements

Connection-State == SAFETY_NBR.

6.4.2.20 SAFETY_QUIT_ERROR

Sends a Quit-Error command to the safe CPU.

```
FUNCTION __cdecl GLOBAL P_Safety_QuietError
VAR_INPUT
    quitRemoteModules : USINT;
END_VAR
VAR_OUTPUT
    retval : DINT;
END_VAR;
#define SAFETY_QUIT_ERROR(pCil,p1) pCil^.QuietError $ P_Safety_QuietError(p1)
```

Transfer parameters	Type	Description					
quitRemoteModules	USINT	Flag that shows whether the error should be canceled in the affected module only or in all removed modules. <table border="1" style="margin-top: 10px;"> <tr> <td style="width: 20px; text-align: center;">0</td><td>only the error in the affected module is canceled</td></tr> <tr> <td style="text-align: center;">≠0</td><td>The safe CPU also sends the Quit-Error command to the removed modules</td></tr> </table> <p>This parameter is only included in the safe CPU starting with Version 337.</p>		0	only the error in the affected module is canceled	≠0	The safe CPU also sends the Quit-Error command to the removed modules
0	only the error in the affected module is canceled						
≠0	The safe CPU also sends the Quit-Error command to the removed modules						
Return parameters	Type	Description					
retval		0	Success				
		≠0	Error code				

Requirements

Connection-State ≥ CS_CONNECTION

6.4.2.21 SAFETY_SET_CONFIGURED

Confirms that the download file was successfully loaded by setting the configuration status in the safe CPU to "Configured + not verified + not distributed".

Before calling this function, [SetUserPromptTime\(\)](#) must be called and a confirmation of the successful transfer requested from the user. The project name, revision number and safe CPU name must thereby be shown. After calling [SetUserPromptTime\(\)](#), this function must then be called within 1 to 60 seconds.

```
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,
```

Transfer parameters		Type	Description	
prjName		^CHAR	Name of the project stored in the download file	
revNbr		^CHAR	Revision number of the project stored in the download file	
scpuName		^CHAR	Name of the safe CPU stored in the download file	
Return parameters		Type	Description	
retval		DINT	0 Success ≠0 Error code	

Requirements

Connection-State == CS_LOGGED_IN und File-State == FS_DOWNLOAD.

6.4.2.22 SAFETY_SET_FILE

Indicates the file created by the Safety Designer for the download.

Sets the file state to FS_File when the file is opened and can be read.

```
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)

```

Transfer parameters	Type	Description	
filename	^CHAR	Name of the file (0-terminated string)	
Return parameters	Type	Description	
retval	DINT	0	OK
		#0	Error code

Requirements

File-State == FS_IDLE

6.4.2.23 SAFETY_SET_IMAGE_FW

Returns the image file for the firmware update.

Sets the file state to FW_FILE, if the file can be opened and read.

```

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)

```

Transfer parameters	Type	Description	
pImagePath	^CHAR	Path to the image on the control (0 terminated string)	
Return parameters	Type	Description	
retval	DINT	0	OK
		#0	Error code

Requirements

File-State == FS_IDLE

6.4.2.24 SAFETY_SET_SAFETY_NBR

Assumes the safety number of the target safe CPU entered by the user. With the connection setup, whether the safety number matches that in the safe CPU.

To call this function, [SetUserPromptTime\(\)](#) must be called and the security number requested by the user. After calling [SetUserPromptTime\(\)](#), this function must then be called within 1 to 60 seconds.

Sets connection state to 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_Safety_SetSafetyNbr(p1)
```

Transfer parameters		Type	Description
safetyNbr		UDINT	Safety number
Return parameters		Type	Description
retval		DINT	0 OK ≠0 Error code

Requirements

Connection-State == IDLE.

6.4.2.25 SAFETY_SET_SERVICE_MODE

Instructs the safe CPU to switch to the runs status SERVICE.

```
FUNCTION __CDECL GLOBAL P_Safety_SetServiceMode
VAR_OUTPUT
    retval : DINT;
END_VAR;
#define SAFETY_SET_SERVICE_MODE(pCil)          pCil^.SetServiceMode $ P_Safety_SetServiceMode()
```

Return parameters		Type	Description
retval		DINT	0 OK

		#0 Error code
--	--	---------------

Requirement

Connection-State == CS_LOGGED_IN.

6.4.2.26 SAFETY_SET_TEMP_SERVICE_MODE

Instructs the safe CPU to temporarily change the run status SERVICE. The application is thereby restarted.

```
FUNCTION __CDECL GLOBAL P_Safety_SetTempServiceMode
VAR_OUTPUT
    retval : DINT;
END_VAR;
#define SAFETY_SET_TEMP_SERVICE_MODE(pCil)      pCil^.SetTempServiceMode $ P_Safety_SetTempServiceMode
```

Return parameters	Type	Description	
retval	DINT	0	OK

Requirements

Connection-State ≥ CS_CONNECTION.

6.4.2.27 SAFETY_SET_USERPROMPT_TIME

Must be called when an input the user required an input. The time point of the call is marked and is then later used to test the plausibility of the input (for example, [SetSafetyNbr\(\)](#) cannot be called directly following [SetUserPromptTime\(\)](#)). If there was an error in the time response, it can be seen in the return value of the following function.

```
FUNCTION __CDECL GLOBAL P_Safety_SetUserPromptTime
VAR_OUTPUT
    retval : DINT;
END_VAR;
#define SAFETY_SET_USERPROMPT_TIME(pCil)      pCil^.SetUserPromptTime $ P_Safety_SetUserPromptTime
```

Return parameters	Type	Description	
-------------------	------	-------------	--

retval	DINT	0 OK #0 Error code
--------	------	-----------------------

6.4.2.28 SAFETY_SET_VERIFIED

Sets the configuration status in the safe CPU to VERIFIED

Before calling this function, [SetUserPromptTime\(\)](#) must be called and a confirmation of the verification process requested from the user. The project name, revision number and safe CPU name must thereby be shown. After calling [SetUserPromptTime\(\)](#), this function must then be called within 1 to 60 seconds.

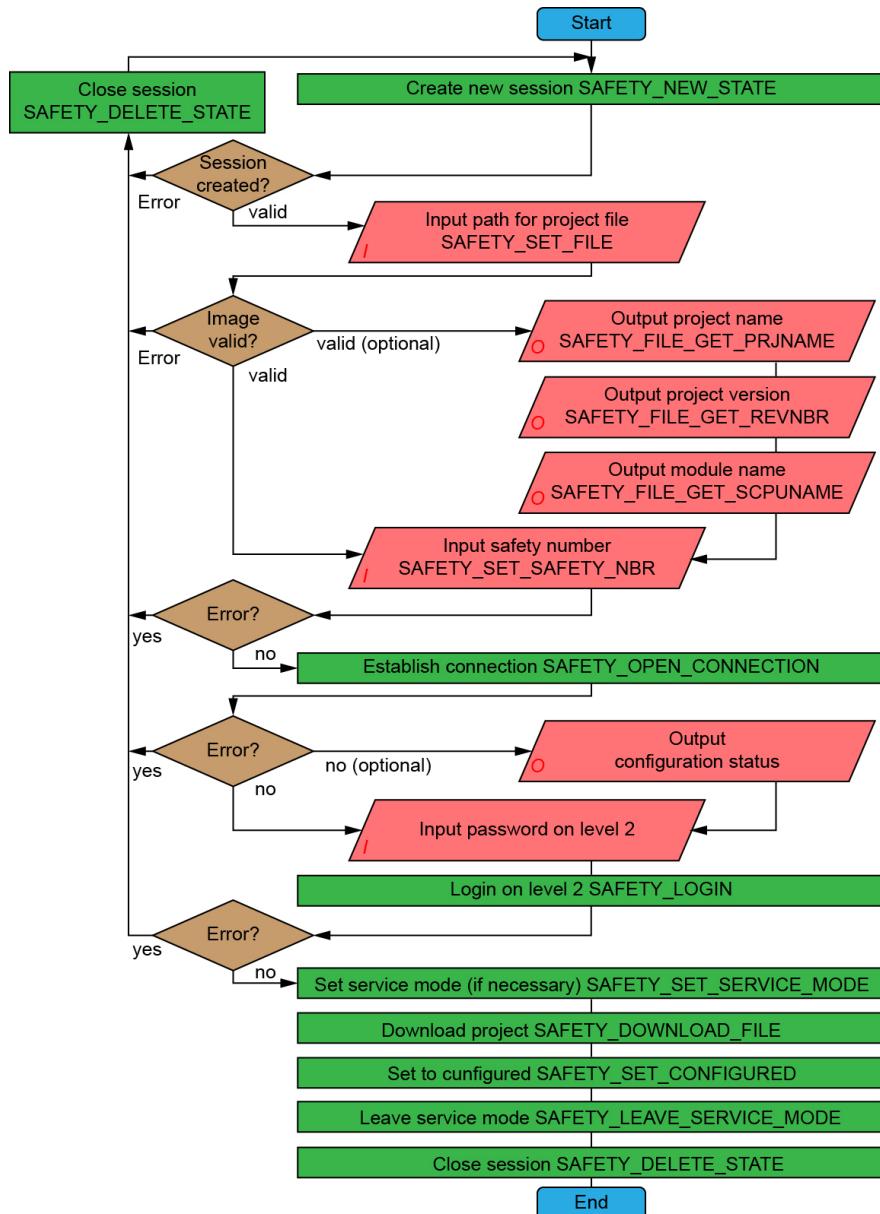
```
FUNCTION __CDECL GLOBAL P_Safety_SetVerified
VAR_INPUT
    prjName    : ^CHAR;
    revNbr    : ^CHAR;
    scpuName   : ^CHAR;
END_VAR
VAR_OUTPUT
    retval    : DINT;
END_VAR;
#define SAFETY_SET_VERIFIED(pCil,p1,p2,p3)    pCil^.SetVerified $ P_Safety_SetVerified(p1,p2,p3)
```

Transfer parameters		Type	Description	
prjName	^CHAR		Name of the project stored in the download file	
revNbr	^CHAR		Revision number of the project stored in the download file	
scpuName	^CHAR		Name of the safe CPU stored in the download file	
Return parameters		Type	Description	
retval	DINT		0 OK #0 Error code	

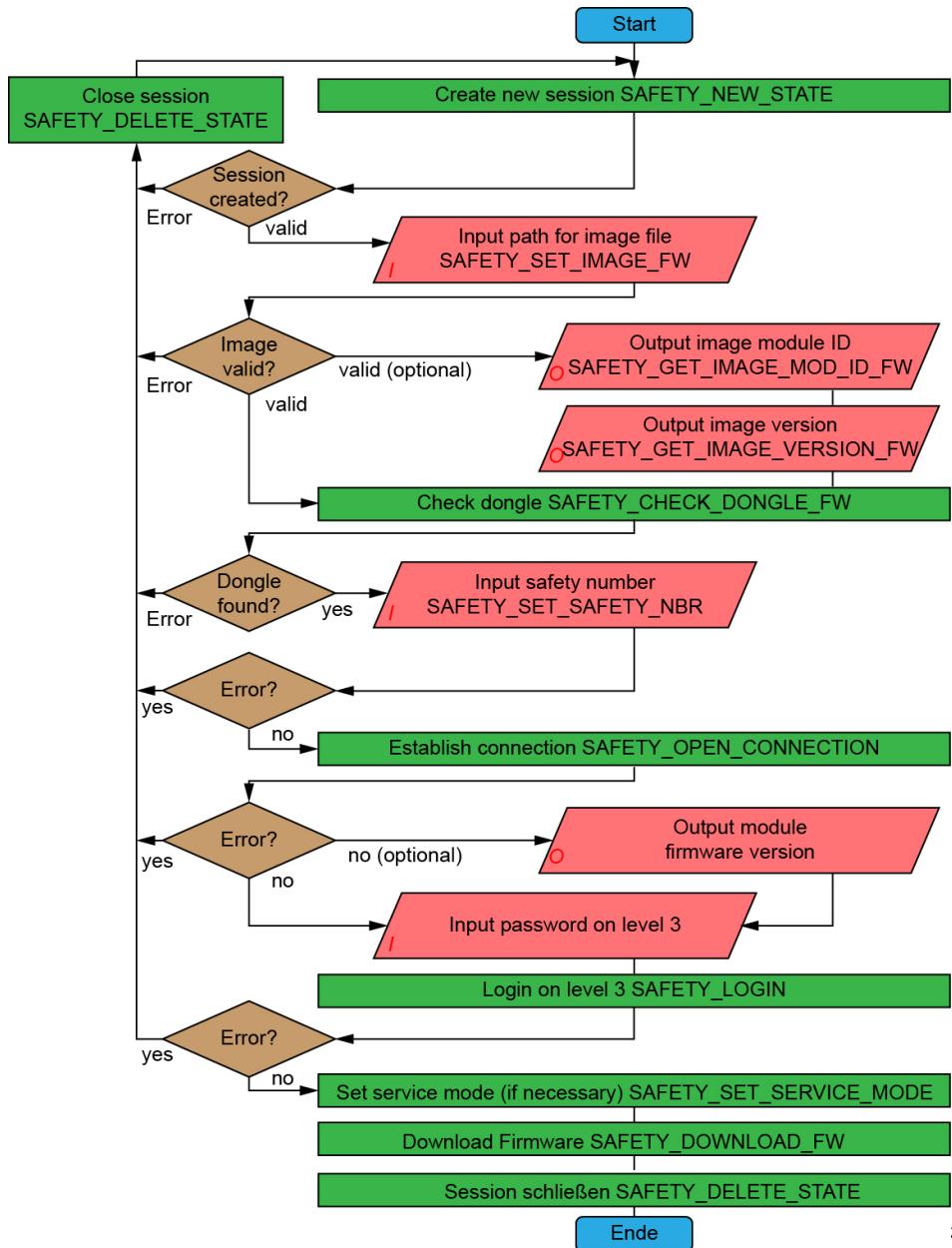
Requirements

Connection-State == CS_LOGGED_IN und File-State == FS_FILE.

6.4.3 Procedure Project Download



6.4.4 Procedure Firmware Download



6.4.5 Error Codes

SAFETY_E_IN_USE	No Safety state could be created (with SAFETY_NEW_STATE), since one already exists.
SAFETY_E_OPEN	When opening the files for download, an error occurred.
SAFETY_E_IO	When reading the files for download, an error occurred.
SAFETY_E_FILEDATA	When reading the files for download, invalid data was detected.
SAFETY_E_INVALID_USERPR OMPT_TIME	A function call, which must be made within a certain time span after calling SAFETY_SET_USERPROMPT_TIME, is made outside of the specified time span.
SAFETY_E_INVALID_CIL_VERS ION	The Safety DLL could not be loaded, because the interface required in the LASAL operating system does not exist or is the wrong version.
SAFETY_E_BUFFER_UNDERFL OW	Too little data was returned with an internal function call.
SAFETY_E_BUF_TOO_SMALL	A buffer assigned as a function parameter is too small.
SAFETY_E_OUT_OF_MEM	Insufficient memory available (Heap).
SAFETY_E_INVALID_SAFETY_ NBR	When creating the connection, it was determined that the security number entered by the user did not match that in the target safe CPU.
SAFETY_E_HWT_CMD	During an internal call of a hardware tree function, an error occurred.
SAFETY_E_INVALID_PARAM	An invalid function parameter value was entered.
SAFETY_E_MODULE_NOT_FO UND	During an attempt to establish the connection, the target module was not found.
SAFETY_E_INVALID_SN_RSP	In the answer received from the safe CPU, an invalid sequence number was detected.
SAFETY_E_INVALID_LEN_RSP	In the answer received from the safe CPU, an invalid length was detected.
SAFETY_E_TOO_LESS_DATA_ RSP	The answer received from the safe CPU contained insufficient data.
SAFETY_E_INVALID_CRC_RSP	In the answer received from the safe CPU, an invalid CRC was detected.
SAFETY_E_INVALID_FRAME TYPE_RSP	In the answer received from the safe CPU, an invalid frame type was detected.
SAFETY_E_INVALID_ADDR_RS P	In the answer received from the safe CPU, an invalid address was detected.
SAFETY_E_TIMEOUT_RSP	The answer expected from the safe CPU was not received within a specified time.

SAFETY_E_INVALID_SESSID_RSP	In the answer received from the safe CPU, an invalid session ID was detected.
SAFETY_E_INVALID_STATE	A function call was performed in an invalid status. The SAFETY_FILE_GET_PRJNAME function must be called for example, after the SAFETY_SET_FILE function.
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 the answer received from the Safe CPU, an error was shown in the return code field. The error number is the offset for the base 10000. E.g. 10123 means that error 123 was returned by the safe CPU.

7 **FTP Server**

7.1 **Instructions**

With the Lasal OS F(ile)T(ransfer)P(rotocol), files can be exchanged between an FTP-Client and the FTP-protocol based server.

7.2 **Quick Start**

The following is a simple description for using the FTP server

7.2.1 **Requirements**

LasalOS	Version 01.01.98 or higher, ftpsvr.dlm version 01.01.001 required
Platform	Any platform with an Ethernet interface

7.2.2 **Installation**

- The FTPSVR.DLM must be copied to the directory C:\LSLSYS
- The default root directory for the FTP-Server is C:\
To change the directory for all clients, the root directory of the server must first be configured.
The server's root directory can be configured using the FTPSVR OPTION command line interface instruction.
E.g.: FTPSVR OPTION DEFROOTDIR C:\FTP-ROOT\
The root directory requires a trailing backslash
Each client can also be assigned its own root directory.
- The FTP-Server can be started with the CLI instruction FTPSVR START
- To add an FTP user, the CLI instruction FTPSVR USER ADD can be used.

7.3 **Security**

7.3.1 User Names

FTP clients require a user name to log into an FTP server (if an anonymous FTP, the username is disabled).

The Lasal S FTP server compares the names, which are not case-sensitive.

E.g.: username FTPUSERX is equivalent to ftpuserx or FTPUserX

7.3.2 Passwords

A password is not mandatory.

A password should always be used for FTP users that have full or at least write access to several files.

Passwords should also be used if only specified users should have access to the server.

7.3.3 Access Rights

 A general access right must be configured for each FTP user.

General access allows entry to all drives, directories and files. In addition to general access, other access rights known as expanded access writes, can also be configured.

With expanded access, access can be granted for a specified drive, directory or file.

For example, a user can have general access R (read-only), expanded access W (write) for the path C:\TEMP and expanded access N (no access) for the file C:\TEMP\text.txt.

In the example the user has read-only access to all drives, directories and files except the directory C:\TEMP and the file C:\TEMP\text.txt.

The user has full access (read/write) to the path C:\TEMP and all subdirectories and files but no access to the file C:\TEMP\text.txt.

 The expanded access rights must be added in the correct order!

This means that the LasalOS FTP server will search the list of access rights starting from the first entry to the last.

E.g.: If C:\TEMP with access 'W' is the first and C:\TEMP\text.txt with access 'N' is the second entry, the user has full access to C:\TEMP\text.txt and the second entry will be ignored.

If FTP user logged into the server wants to overwrite the C:\TEMP\text.txt file, the LasalOS FTP server searches the list of expanded access rights to find the entry C:\TEMP with access W.

Enlarged access rights order:

C:\TEMP
C:\TEMP\text.txt

The entry C:\TEMP\text.txt will be ignored.

Correct order:

C:\TEMP\text.txt
C:\TEMP

7.3.4 Anonymous FTP

The Lasal OS FTP server supports anonymous FTPs.

By default, the FTP server is non-anonymous.

If an anonymous FTP is enabled, usernames and passwords are not required to log into the FTP server.

If the anonymous FTP option is enabled, the expanded anonymous FTP access is available for anonymous FTP clients.



If the anonymous FTP access right is set to write access ('W'), all anonymous clients have full access to all drives, files and directories.

7.4 Server Configuration

Several options are available for changing the configuration of the Lasal OS FTP server.

7.4.1 ANONYMOUS

Option for enabling or disabling an anonymous FTP.

Values

0	OFF: disables anonymous FTP (default)
1	ON: enables anonymous FTP

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.2 ANONYMACCESS

Option for setting the access right for anonymous FTP clients.

Values

N	FTP_ACCESS_NONE: no access
R	FTP_ACCESS_READ: read-only access (default)
W	FTP_ACCESS_WRITE: full access (read/write/delete)

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.3 NOFTPSINI

Option to enable or disable writing FTP users to the ftps.ini file. If this option is ON, all registered FTP users will be lost when the server is shut down (stop) or the system rebooted.

Values

0	OFF: ftps.ini will be used (default)
1	ON: ftps.ini will not be used

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.4 WINCOMPLISTFORMAT

This option can be used to change the format of the data content sent using a list command.

Values

0	OFF: Default list format (default)
1	ON: Microsoft Windows compatible list format

This option can be activated to show the time information of files and folders when using the Microsoft Internet Explorer for an FTP session or any compatible FTP client.

Since it varies from the standard FTP format specification, not every FTP client is required to handle the Microsoft Windows compatible list format.

According to the FTP standard format specification, no time information will be sent to the FTP client when the date information contains the year, but it could be sent instead of the year or as a combination of both, or the year and time information individually. The Lasal OS FTP server is set by default to send the date information with the year.

Supported FTP Commands

FTP command	DLL Version	Brief description
QUIT	01.01.001	Logout from server, terminate user

HELP	01.01.001	Send information with available commands to the client
NOOP	01.01.001	No operation
USER	01.01.001	Username to login to server
PASS	01.01.001	The users password
MODE	01.01.001	Transfer mode (S-Stream, B-Block, C-Compressed), the FTP-Server supports S only
TYPE	01.01.001	Representation type (A,I)0
STRU	01.01.001	Specifies file structure (F-File, R-Record structure, P-Page structure), the FTP-Server supports F only
PORT	01.01.001	Specification for the data port to be used in data connection
PASV	01.01.001	Scans a different data port from the default
SYST	01.01.001	Determine the type of operating system
DELE	01.01.001	Delete a file
RMD	01.01.001	Remove a directory
CDUP	01.01.001	Change to parent directory
CWD	01.01.001	Change current working directory
MKD	01.01.001	Create a directory
PWD	01.01.001	Returns the name of the current working directory
RNFR	01.01.001	Rename from
RNTO	01.01.001	Rename to
SIZE	01.01.001	Size of a file
LIST	01.01.001	Directory list
NLST	01.01.001	Directory name list
STOR	01.01.001	Accept data and store it as a file
RETR	01.01.001	Send a copy of a file
APPE	01.01.001	Accept data and store/append it as/to a file

Requirements

LasalOS: 01.01.098 or higher to set the option via the method [SetConfig](#) of the `_FTPServer` class (see chapter [Lasal OS FTP-Server API](#)) or 01.02.004 to set the option

via the CLI command [FTPSVR_OPTION](#) (see chapter [Command Line Interface \(CLI\) Commands](#))

FTP-Server DLL version 01.01.004 or higher

7.4.5 PASVPORT

With this option, the port value that should be used by the FTP server with a passive data connection can be set.

Values

0-65535 (default: 0, dynamic)

If the value 0 is used, the FTP server will allocate a port dynamically

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.6 DEFROOTDIR

Option for setting the default root directory the FTP server will use when a client doesn't use it's own root directory.

Values

A null-terminated string (default: C:\)

The root directory requires a trailing backslash.



Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.7 MAXSESSIONS

Option for setting the maximum number of sessions the FTP server will allow simultaneously.

Values

- 0 (default: 0, no limit, DLL version 01.01.001)
(default: 10, DLL version > 01.01.001)
If the value is 0, no limit is set

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.8 FORWARDSLASH

If this option is enabled, the FTP server converts all backslashes to forward slashes when returning paths for a PWD to a client.

Values

- 0 OFF: do not convert (default)
- 1 ON: converts back to forward slashes

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.9 BACKSLASH

If this option is enabled, the FTP server converts all forward slashes to back slashes in file name received from a client.

Values

- 0 OFF: do not convert

1	ON: converts forward to back slashes (default)
---	------------------------------------------------

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.10 NODRIVE

If this option is enabled, the FTP server removes the drive from the path with a PWD command.

Values

0	OFF: do not remove
1	ON: remove drive letter (default)

If the drive letter and colon are not removed, the server is not compatible with all Unix-based clients.

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.11 CMDTMO

Option to change the command timeout.

Wait time for a command from the client.

Values

0-60	FTPSVR_INF (default: 30)
------	--------------------------

Unit

seconds

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.12 WRITETMO

Option for changing the write timeout.

Wait time for sending data to remote hosts.

Values

0-60 **FTPSVR_INF** (default: 30)

Unit

seconds

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.13 READTMO

Option for changing the read timeout.

Wait time for receiving data (files) and acknowledgments from a remote host.

Values

0-60 **FTPSVR_INF** (default: 30)

Unit

seconds

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.14 TPrio

Option for changing the connection task priority.

Values

1-14 (default: 8)

This option should be used very carefully.

The priority of the connection task can be the same as the FTP server daemon task priority but not higher.



Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.4.15 DPrio

Option to change the priority of the FTP server daemon task (the main task of the server that scans for incoming connections)

Values

1-14 (default: 9)

This option should be used with caution.

The default setting of the FTP server daemon task priority is lower than that of the background task. If the background task the priority does not allow enough time for of the FTP server daemon task, the priority of the daemon task should be increased to that of the background task.



Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.5 Command Line Interface (CLI) Commands

7.5.1 FTPSVR INFO

Provides information on the FTP server, the status, registered users, active connections, the DLL Version and the actual FTP server configuration.

Requirements

LasalOS: 01.01.098 or higher, FTP server DLL version 01.01.001 or higher

7.5.2 FTPSVR HELP

Provides information on all available FTP server CLI commands and their use.

Requirements

LasalOS: 01.01.098 or higher, FTP server DLL version 01.01.001 or higher

7.5.3 FTPSVR START

Starts the FTP server. This command can be executed directly through the command line interface or the autoexc.lsl.

When the FTP server is started, the ftps.ini file containing all previously logged in and saved FTP users is loaded.

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.5.4 FTPSVR OPTION

This command is used to change or display the FTP server configuration

If no parameter is given, the command displays the actual configuration.

Optional Parameters

<option> Option to configure

<value> Value for the specified option

<help> Lists all available options and their possible values

See the chapter on “Server Configuration” for available options and their possible values. If the parameter option is given without a value, the current value of the option will be displayed.

Example

List of actual FTP-Server configuration.

```
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:\>
```

Actual configuration for option anonymous FTP.

```
C:\> FTPSVR OPTION ANONYMOUS
Anonymous FTP .....: OFF
```

```
C:\>
```

Enable anonymous FTP.

```
C:\> FTPSVR OPTION ANONYMOUS ON
```

```
C:\>
```

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher.

7.5.5 FTPSVR ACCESS

This command adds and removes access rights for FTP clients.

Optional and required arguments:

ADD <username>

Adds an access right for a specified FTP user <username> and stores it to the `ftps.ini` file.

<username>	FTP user for which the access right is added
------------	----------------------------------------------

Example

```
C:\> FTPSVR ACCESS ADD FTPuser1
```

```
Enlarged 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	Specified path for which the following access right is valid
Access right	Access right for the above path
Insert at position	Position on which the entry should be inserted (no position to append the entry)

See the chapter on SECURITY for more information on using access rights.

The newly added access right entry is saved to the `ftps.ini` file when the FTP server option `NOFTPSINI` is OFF (default). The saved FTP users and their access rights are loaded with the next server startup.

REMOVE <username> <path>

Removes an access right given by the path of a specified FTP user.

<username>	FTP user for which the access right should be removed
------------	-------------------------------------------------------

<path>	Path of the access right that should be removed
--------	-------------------------------------------------

Example

C:\> FTPSVR ACCESS REMOVE FTPuser1 C:\LSLSYS

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher.

7.5.6 FTSPSVR USER

This command adds, removes and gives information of FTP clients.

Optional and required arguments:

INFO <username> </p>

Shows information of FTP users.

If no parameter is given, the command shows the information of all registered FTP clients.

Optional Parameters

<username>	Shows information of FTP user <username>
------------	------------------------------------------

</p>	Stops output after every full screen page
------	-------------------------------------------

Example

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
Enlarged access rights	

[W] ... C:\TEMP\

C:\>

This command cannot be executed from the autoexec.lsl.

ADD

The ADD command is used to add an FTP user and has no parameters. The user is prompted to enter the information needed to add an FTP user after executing the command. This command cannot be executed from the autoexec.lsl

Example

```
C:\> FTPSVR USER ADD
```

```
Username (max. 63).....: FTPuser1
Password (max. 63).....: test123
General Access (N,R,W): W
Root directory.....:
```

```
FTP user created successfully
```

```
Enlarged access rights: No path to exit, no position to append entry
```

```
Access path.....:
```

```
Saving FTP user... successful
```

```
C:\>
```

username	User name, which the FTP client can use to connect to the server (the following characters are not allowed: '/', '\')
Password	Password that is needed to login with a specified user. A password is not mandatory; if an FTP user should be able to login without a password, none must be entered.
General Access	General access for the FTP client (this kind of access is valid for all drives, directories and files)
Root directory	Root directory that is used for the FTP client. The servers default root directory is used if no root directory is specified.

For expanded access rights, see FTPSVR ACCESS

The newly added entry is saved to the `ftps.ini` file when the FTP server option `NOFTPSINI` is OFF (default). The saved FTP users are then loaded with the next server start-up.

REMOVE <username>

This command removes an FTP user and deletes it from the `ftps.ini` file.

<username>	FTP user to remove (required parameter)
------------	-----------------------------------------

Example

```
C:\> FTPSVR USER REMOVE FTPuser1
```

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher.

7.5.7 FTSPSVR STOP

Stops the FTP server and disconnects all active FTP clients.

Requirements

LasalOS: 01.01.098 or higher, FTP-Server DLL version 01.01.001 or higher

7.6 Lasal OS FTP-Server API

7.6.1 OS Interface Library Class _FTPServer

The operating system interface is available through the _FTPServer class in the OS Interface library. The OS Interface library is available for LASAL Class 1 and LASAL Class 2.

To use the interface, a new project must be created and the OS Interface library class _FTPServer imported. A new class with an object channel to the _FTPServer class must be created.

The methods of the interface can be called through the object channel created

Example

Object channel for the newly created class: ocFTPSvr

Call a method: ocFTPSvr.methodname(parameter, ...)

7.6.1.1 AddAccess

Adds an expanded access right to an existing FTP user.

```
FUNCTION __CDECL VIRTUAL GLOBAL AddAccess
VAR_INPUT
    Username    : ^CHAR;
    Access      : UDINT;
    Path        : ^CHAR;
    InsertPos   : UDINT;
END_VAR
VAR_OUTPUT
    dRC (EAX)   : DINT;
END_VAR;
```

Transfer parameters		Type	Description
Username		^CHAR	Null-terminated string containing the name of the FTP user to which an expanded access right should be added
Access		UDINT	The access right FTP_ACCESS_NONE - no access FTP_ACCESS_READ - read-only access FTP_ACCESS_WRITE - full access (read/write/delete)
Path		^CHAR	Null-terminated string containing the path or file name for which the access right given by the access parameter is valid. The access right is valid for the path and all files and directories that the path contains. A trailing backslash is not required for these paths.
InsertPos		UDINT	The position in which the access right should be added. 1 = first, 2 = second, ... 0 to add entry
Return parameters		Type	Description
dRC		DINT	0 Success <0 Negative error code

An access right cannot be added if an FTP client is logged in with the name as the FTP user entry access right that should be expanded.

If the FTP user was added using the method [AddUser\(\)](#) and the Save parameter was set to 1, the added access rights can be saved to the `ftps.ini` file by calling the method [UpdateFTPSini\(\)](#).

See the chapter on security access rights for more information of access rights.

Example

```

dRC : DINT;

// add expanded access rights for the user "FTPUser"

// full access to directory "C:\TEMP", but read-only access for the file
// "C:\TEMP\test.txt"
dRC := ocFTPSvr.AddAccess("FTPUser", FTP_ACCESS_READ, "C:\TEMP\test.txt",0);
if dRC = 0 then
    // access right successfully added

    // add next
    dRC := ocFTPSvr.AddAccess("FTPUser", FTP_ACCESS_WRITE, "C:\TEMP",0);
    if dRC = 0 then
        // access right successfully added
    end_if;
end_if;
if dRC <> 0 then
    // error occurred
end_if;

```

7.6.1.2 AddUser

Adds a FTP user to the registered list of available FTP users.

```

FUNCTION __CDECL VIRTUAL GLOBAL AddUser
VAR_INPUT
    Username      : ^CHAR;
    Password      : ^CHAR;
    GeneralAccess : UDINT;
    RootDir       : ^CHAR;
    Save          : USINT;
END_VAR
VAR_OUTPUT
    dRC (EAX)      : DINT;
END_VAR;

```

Transfer parameters	Type	Description
Username	^CHAR	0-terminated string containing the user name to add, max. FTPSVR_USERNAME_LEN characters
Password	^CHAR	0-terminated string containing the password, max. FTPSVR_PASSWORD_LEN characters. If no password is desired, a NIL pointer can be assigned.
GeneralAccess	UDINT	The general access right for the FTP user FTP_ACCESS_NONE - no access FTP_ACCESS_READ - read-only access

		FTP_ACCESS_WRITE - full access (read/write/delete)				
RootDir	^CHAR	0-terminated string containing the root directory for the FTP user with a max of 260 characters. If no separate root directory is needed, a NIL pointer can be assigned. In such a case, the default root directory from the server is used.				
Save	USINT	<p>Sets a flag to store the entry to the ftps.ini file by calling the method UpdateFTPSini().</p> <table border="1" style="margin-left: 20px;"> <tr> <td>0</td> <td>does not store the entry</td> </tr> <tr> <td>1</td> <td>stores the entry</td> </tr> </table>	0	does not store the entry	1	stores the entry
0	does not store the entry					
1	stores the entry					
Return parameters	Type	Description				
dRC	DINT	<table border="1" style="margin-left: 20px;"> <tr> <td>0</td> <td>Success</td> </tr> <tr> <td><0</td> <td>Negative error code</td> </tr> </table>	0	Success	<0	Negative error code
0	Success					
<0	Negative error code					

When the method is executed successfully, an FTP client can login to the FTP server with the name and password of the newly added FTP user.

If the Save parameter is 0, the FTP user entry will be removed when the application is reset. If the parameter is 1, the user will not be removed with an application reset. To store the entry so that it is available with the next FTP Server startup or reboot, the [UpdateFTPSini\(\)](#) method must be called.

The general access right is valid for all drives, directories and files. If the GeneralAccess parameter is FTP_ACCESS_WRITE, the user has full access to all drives, directories and files.

Expanded access rights can be set for drives, directories and files. (see [AddAccess\(\)](#))

Following characters are not allowed in a username: /, \

Example

```

dRC      : DINT;

dRC := ocFTPSvr.AddUser("FTPUser", "123", FTP_ACCESS_READ, NIL, 0);
if dRC = 0 then
  // add user done
else
  // add user failed
end_if;

```

7.6.1.3 EditAccess

This instruction changes an existing expanded access right.

```
FUNCTION __CDECL VIRTUAL GLOBAL EditAccess
VAR_INPUT
    Username    : ^CHAR;
    Access      : UDINT;
    Path        : ^CHAR;
    newPath     : ^CHAR;
    InsertPos   : UDINT;
END_VAR
VAR_OUTPUT
    dRC (EAX)   : DINT;
END_VAR;
```

Transfer parameters	Type	Description	
Username	^CHAR	0-terminated string containing the user name for the FTP user to which an expanded access right should be added	
Access	UDINT	The access right FTP_ACCESS_NONE - no access FTP_ACCESS_READ - read-only access FTP_ACCESS_WRITE - full access (read/write/delete)	
Path	^CHAR	0-terminated string containing the path or file name for which the access right given by the access parameter is valid. The access right is valid for the path and all files and directories the path contains. A trailing backslash is not required for paths	
NewPath	^CHAR	0-terminated string to a new path, if this parameter is a NIL pointer, the path remains unchanged	
InsertPos	UDINT	The position on which the access right should be added. 1 = first, 2 = second, ...0 to append the entry	
Return parameters	Type	Description	
dRC	DINT	0 User was changed successfully FTPSVR_ERR_EXISTS <0 Negative error code	No changes to the existing user

An access right cannot be changed if an FTP client is logged in with the name as the FTP user entry access right that should be changed.

If the FTP user was added using the method [AddUser\(\)](#) and the Save parameter was set to 1, the changed access rights can be saved to the `ftps.ini` file by calling the method [UpdateFTPSini\(\)](#).

Only existing access rights can be changed.

The parameters must be different from the existing access right in order to change an access right entry, otherwise the error `FTPSVR_ERR_EXISTS` is returned.

If only the Access parameter is different from the previously added or changed access right and the path is the same or a NIL pointer has been assigned to indicate that no changes for the path of the access right have been specified, the parameter `InsertPos` is ignored.

Example

```
dRC : DINT;

// edit existing access right (C:\TEMP, full access)
dRC := ocFTPSvr.EditAccess("FTPUser", FTP_ACCESS_READ, "C:\TEMP", 0);
if dRC = 0 then
  // access right successfully added
else
  // error occurred
end_if;
```

7.6.1.4 EditUser

This instruction is used to change an existing FTP user added using the method [AddUser\(\)](#).

```
FUNCTION __CDECL VIRTUAL GLOBAL EditUser
VAR_INPUT
  Username      : ^CHAR;
  NewUsername   : ^CHAR;
  Password      : ^CHAR;
  GeneralAccess : UDINT;
  RootDir       : ^CHAR;
  Flags         : UDINT;
END_VAR
VAR_OUTPUT
  dRC (EAX)      : DINT;
END_VAR;
```

Transfer parameters	Type	Description
Username	^CHAR	Null-terminated string containing the name of the FTP user that should be changed. The maximum length is determined by <code>FTPSVR_USERNAME_LEN</code> .

NewUsername	^CHAR	Null-terminated string containing a new user name, if this parameter is a NIL pointer, the user name remains unchanged.						
Password	^CHAR	Null-terminated string containing the password. The maximum length is determined by FTPSVR_PASSWORD_LEN. If no password is desired, a NIL-pointer can be assigned.						
GeneralAccess	UDINT	The general access right for the FTP user FTP_ACCESS_NONE - no access FTP_ACCESS_READ - read-only access FTP_ACCESS_WRITE - full access (read/write/delete)						
RootDir	^CHAR	Null-terminated string containing the root directory for the FTP user with a maximum of 260 characters. If no separate root directory is needed, a NIL pointer can be assigned. In such a case, the default root directory from the server will be used.						
Flags	UDINT	Specifies which parameters should be changed FTPSVR_CHANGE_PASSWORD - change password FTPSVR_CHANGE_GENERALACCESS - change general access FTPSVR_CHANGE_ROOTDIR - change root directory						
Return parameters	Type	Description						
dRC	DINT	<table border="1"> <tr> <td>0</td><td>User was changed successfully</td></tr> <tr> <td>FTPSVR_ERR_EXISTS</td><td>No changes to the existing user</td></tr> <tr> <td><0</td><td>Negative error code</td></tr> </table>	0	User was changed successfully	FTPSVR_ERR_EXISTS	No changes to the existing user	<0	Negative error code
0	User was changed successfully							
FTPSVR_ERR_EXISTS	No changes to the existing user							
<0	Negative error code							

Only existing FTP users can be changed.

The parameters must be different from the existing FTP user in order to change an FTP user entry independent of the parameter Flags, otherwise the error FTPSVR_ERR_EXISTS will be returned.

If the FTP user was added using the [AddUser\(\)](#) method and the parameter Save is set to 1, the changes can be saved to the `ftps.ini` file by calling the method [UpdateFTPSini\(\)](#).

An FTP user cannot be changed if an FTP client is logged in with the name of the FTP user entry that should be changed.

Example

```
dRC      : DINT;
// change the username of an existing entry
```

```

dRC := ocFTPSvr>EditUser("FTPUser", "NewFTPUser", NIL, 0, NIL, 0);

// no flags given, password, general access, root directory remains
// unchanged

if dRC = 0 then
  // FTP user successfully changed
else
  // error occurred
end_if;

// change the password of the above user, this could also be done
// by calling the EditUser method
dRC := ocFTPSvr>EditUser("NewFTPUser",
  NIL,
  "456",
  0,
  NIL,
  FTPSVR_CHANGE_PASSWORD) ;

if dRC = 0 then
  // FTP user successfully changed
else
  // error occurred
end_if;

```

7.6.1.5 GetConfig

This function reads an option from the actual FTP server configuration.

```

FUNCTION __CDECL VIRTUAL GLOBAL GetConfig
VAR_INPUT
  pValue      : pVoid;
  Size        : UDINT;
  Option      : UDINT;
END_VAR
VAR_OUTPUT
  dRC (EAX) : DINT;
END_VAR;

```

Transfer parameters		Type	Description
pValue	pVoid		Pointer to variable to retrieve the value of the specified option
Size	UDINT		Size of the variable, array or memory area in bytes to which pValue points
Option	UDINT		The option to get (see chapter Appendix A – Types for a list of available option)
Return parameters		Type	Description

dRC	DINT	0 Success	
		<0 Negative error code	

The following is a list of the available options and the size for each option to retrieve the value.

Option	Type of 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

The size for option **FTPS_OPT_DEF_ROOT_DIR** is variable but must be at least the length of the default root directory in bytes plus 1 byte for the terminating zero.

The root directory can be up to 260 characters (bytes).

Example

```

dRC          : DINT;
anonymousFTP : USINT;
RootDir      : ARRAY [0..26] OF CHAR;

// get option ANONYMOUS (anonymous FTP)

// use parameter pValue
dRC := ocFTPSvr.GetConfig(#anonymousFTP, 0,

```

```
dRC := ocFTPSvr.SetConfig("ON", 0, FTPS_OPT_ANONYMOUS);

// get default root directory
dRC := ocFTPSvr.SetConfig("C:\FTPROOT", 0, FTPS_OPT_DEF_ROOT_DIR);
```

7.6.1.6 GetUpdateFTPSiniStatus

This command retrieves the status of the write process for the `ftps.ini` file when using the [UpdateFTPSini\(\)](#) method in non-blocking mode.

```
FUNCTION __CDECL VIRTUAL GLOBAL GetUpdateFTPSiniStatus
VAR_INPUT
    pRC      : ^DINT;
END_VAR
VAR_OUTPUT
    dRC (EAX) : DINT;
END_VAR;
```

Transfer parameters		Type	Description	
Return parameters		Type	Description	
pRC	^DINT		0	Method finished
dRC	DINT	FTPSVR_ERR_UPD_BUSY	Write process is in progress	Method is called again after it has returned 0

The method should be called as long as the return value is `FTPSVR_ERR_UPD_BUSY`.
 The content of the `DINT` value to which `pRC` points, remains unchanged until the internal task is ready and the method returns 0.

Example

```
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

This method retrieves the expanded access rights for a specified FTP user.

```

FUNCTION __CDECL VIRTUAL GLOBAL GetUserAccessInfo
VAR_INPUT
  Username    : ^CHAR;
  Path        : ^CHAR;
  SizeOfPath  : UDINT;
  Access      : ^UDINT;
  AccessIdx   : UDINT;
END_VAR
VAR_OUTPUT
  dRC (EAX)  : DINT;
END_VAR;

```

Transfer parameters	Type	Description
Username	^CHAR	0-terminated string containing the name of the FTP user for which the extended information is retrieved
Path	^CHAR	Pointer to an array or allocated memory area that retrieves the path of the FTP users expanded access rights
SizeOfPath	UDINT	Size of the array or memory area to which Path points
Access	^UDINT	Pointer to a variable of type UDINT that retrieves the value for the access right
AccessIdx	UDINT	Index of the FTP users access right to receive (start with 0)
Return parameters	Type	Description
dRC	DINT	0 FTP user was found successfully

		FTPSVR_ERR_NOT_FOUND	FTP user does not exist or no access rights are available
--	--	----------------------	-----------------------------------------------------------

To retrieve the expanded access right for all FTP users, the method must be called until **FTPSVR_ERR_NOT_FOUND** is returned.

With each call, the **AccessIdx** parameter must be incremented by one.

An **AccessIdx** of 0 will return the information of the first access right of an FTP user.

To retrieve the complete information for all FTP users, the [**GetUserInfo\(\)**](#) method should first be called to retrieve the first FTP user. If an FTP user was found, the methods [**GetUserExtendedInfo\(\)**](#) and [**GetUserAccessInfo\(\)**](#) can then be called to retrieve the rest of the FTP user information.

Once the [**GetUserInfo\(\)**](#) method can be called, the **UserIdx** parameter can then be incremented by one to search for the next FTP user.

Example

```

dRC      : DINT;
Path     : ARRAY [0..260] OF CHAR;
Access   : UDINT;

// get first enlarged access rights for FTP user "FTPUser"
dRC = ocFTPSvr.GetUserAccessInfo("FTPUser",
                                  #Path[0],
                                  sizeof(Path),
                                  #Access,
                                  0);

if dRC = 0 then

  // get second enlarged access rights for FTP user "FTPUser"
  dRC = ocFTPSvr.GetUserAccessInfo("FTPUser",
                                  #Path[0],
                                  sizeof(Path),
                                  #Access,
                                  0);

  // ... until GetUserAccessInfo return FTPSVR_ERR_NOT_FOUND

end_if;

```

7.6.1.8 GetUserExtendedInfo

This command retrieves extended information from a specified FTP user.

```
FUNCTION __CDECL VIRTUAL GLOBAL GetUserExtendedInfo
VAR_INPUT
    Username      : ^CHAR;
    pValue        : pVoid;
    SizeOfValue   : UDINT;
    UserInfo      : UDINT;
END_VAR
VAR_OUTPUT
    dRC ( EAX )   : DINT;
END_VAR;
```

Transfer parameters	Type	Description	
Username	^CHAR	0-terminated string containing the name for the FTP user from which the extended information is retrieved	
pValue	pVoid	Pointer to the variable that retrieves the value of the extended information	
SizeOfValue	UDINT	Size of the variable, array or memory area in bytes to which pValue points	
UserInfo	UDINT	The type of extended information retrieved FTPS_USR_INFO_REFERENCE FTPS_USR_INFO_STATUS FTPS_USR_INFO_LOGGEDIN	
Return parameters	Type	Description	
dRC	DINT	0 Success <0 Negative error code	

The following table shows the available extended information, the data type for the parameter pValue, and their size.

UserInfo	Type	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

See [Appendix A – Types](#) for the values and description for all extended information.

Example

```

dRC : DINT
ref : USINT;
stat : UDINT;

// get extended info "Reference"
dRC := ocFTPSvr.GetUserExtendedInfo("FTPUser",
                                      #ref,
                                      sizeof(ref),
                                      FTPS_USR_INFO_REFERENCE);

// get extended info "Status"
dRC := ocFTPSvr.GetUserExtendedInfo("FTPUser",
                                      #ref,
                                      sizeof(ref),
                                      FTPS_USR_INFO_STATUS);

```

7.6.1.9 GetUserInfo

This method is used to retrieve all registered FTP users, their general access right and the root directory.

```

FUNCTION __CDECL VIRTUAL GLOBAL GetUserInfo
VAR_INPUT
  Username      : ^CHAR;
  SizeOfUsername : UDINT;
  GeneralAccess  : ^UDINT;
  RootDir        : ^CHAR;
  SizeOfRootDir  : UDINT;
  UserIdx        : UDINT;
END_VAR
VAR_OUTPUT
  drc (EAX)      : DINT;
END_VAR;

```

Transfer parameters	Type	Description
Username	^CHAR	Pointer to an array or allocated memory area that retrieves the name of the FTP user
SizeOfUsername	UDINT	Size of the array or memory area in bytes to which Username points
GeneralAccess	^UDINT	Pointer to a variable of type UDINT to receive the value for the general access
RootDir	^CHAR	Pointer to an array or allocated memory area to retrieves the root directory of the FTP user
SizeOfRootDir	UDINT	Size of the array or memory area in bytes to which RootDir points
UserIdx	UDINT	Index of the FTP user to retrieve (start with 0)

Return parameters	Type	Description	
dRC	DINT	0	Success and FTP user was found
		FTPSVR_ERR_NOT_FOUND	No FTP users are available
		NOT_FOUND	

To retrieve the information from all FTP users, the method must be called until it returns **FTPSVR_ERR_NOT_FOUND**. With each call, the parameter **UserIdx** must be incremented by one. A **UserIdx** of 0 will return the information of the first FTP user.

Example

```

Username      : 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

  // second user
  dRC := ocFTPSvr.GetUserInfo(#Username[0],
                           sizeof(Username),
                           #GeneralAccess,
                           #RootDir[0],
                           sizeof(RootDir),
                           0);

  // ... until GetUserInfo return FTPSVR_ERR_NOT_FOUND

end_if;

```

7.6.1.10 RemoveAccess

This instruction removes an expanded access right.

```

FUNCTION __cdecl virtual GLOBAL RemoveAccess
VAR_INPUT
  Username  : ^CHAR;
  Path      : ^CHAR;

```

```

END_VAR
VAR_OUTPUT
    dRC (EAX) : DINT;
END_VAR;

```

Transfer parameters		Type	Description	
Username		^CHAR	0-terminated string containing the name of the FTP user for which the access right should be removed	
Path		^CHAR	0-terminated string containing the path for the access right to remove	
Return parameters		Type	Description	
dRC		DINT	0 Success <0 Negative error code	

An existing access right cannot be if an FTP client is logged in with the same name as the FTP user for which the access right should be removed.

If the FTP user was added using the [AddUser\(\)](#) method and the Save parameter is set to 1, it is possible to remove the access right in the ftps.ini file calling the [UpdateFTPSini\(\)](#) method.

Example

```

dRC : DINT;

dRC := ocFTPSvr.RemoveAccess("FTPUser", "C:\TEMP");
if dRC = 0 then
    // access right successfully added
else
    // error occurred
end_if;

```

7.6.1.11 RemoveUser

Removes an existing FTP user.

```

FUNCTION __CDECL VIRTUAL GLOBAL RemoveUser
VAR_INPUT
    Username : ^CHAR;
END_VAR
VAR_OUTPUT
    dRC (EAX) : DINT;
END_VAR;

```

Transfer parameters		Type	Description	
---------------------	--	------	-------------	--

Username		Null-terminated string containing the user name to remove
Return parameters	Type	Description
dRC	DINT	0 Success <0 Negative error code

It is not possible to remove an FTP user if an FTP client is logged in with the name of the FTP user entry that should be removed.

The method does not remove the FTP user from the `ftps.ini` file.

To update the `ftps.ini` file, the [UpdateFTPSini\(\)](#) method must be called

Example

```
dRC : DINT;

// change the username of an existing entry
dRC := ocFTPSvr.RemoveUser("NewFTPUser");
if dRC = 0 then
  // FTP user successfully changed
else
  // error occurred
end_if;
```

7.6.1.12 StartServer

This method starts the FTP server, if not already running.

```
FUNCTION __CDECL VIRTUAL GLOBAL StartServer
VAR_OUTPUT
  dRC (EAX) : DINT;
END_VAR;
```

Transfer parameters	Type	Description
none		
Return parameters	Type	Description
dRC	DINT	0 Success <0 Negative error code

If the FTP-Server is already running the return value is `FTPSVR_ERR_EXISTS`.

The method reads the data from the `ftps.ini` file and creates the FTP users stored in the file.

The amount of time required by the method depends on the size of the `ftps.ini` file. This method should therefore, not be called in cyclic or real-time tasks.

Example

```
dRC : DINT;

dRC := ocFTPSvr.StartServer();
if dRC = 0 then
  // start server done
else
  // start server failed
end_if;
```

7.6.1.13 StopServer

Stops the FTP server and disconnects all active connected and logged in FTP clients.

```
FUNCTION __CDECL VIRTUAL GLOBAL StopServer
VAR_OUTPUT
  dRC (EAX) : DINT;
END_VAR;
```

Transfer parameters	Type	Description
none		
Return parameters	Type	Description
dRC	DINT	0 Success <0 Negative error code

All created FTP users not stored in the `FTPS.INI` file are lost when this method is called. All active connected and logged in clients will be disconnected from the server whether or not the data transfer is completed. The amount of time the method requires depends on the number of active clients connected, which must then be disconnected and the number of registered FTP users. This method should therefore, not be called in cyclic or real-time tasks.

Example

```
dRC : DINT;

dRC := ocFTPSvr.StopServer();
if dRC = 0 then
  // stop server done
else
```

```
// stop server failed
end_if;
```

7.6.1.14 SetConfig

This command is used to change the actual configuration of the FTP server.

```
FUNCTION __CDECL VIRTUAL GLOBAL SetConfig
VAR_INPUT
  pValue      : ^CHAR;
  udValue     : UDINT;
  Option      : UDINT;
END_VAR
VAR_OUTPUT
  dRC (EAX) : DINT;
END_VAR;
```

Transfer parameters	Type	Description				
pValue	^CHAR	Null-terminated string containing the value for the option to set				
udValue	UDINT	Variable that contains the value for the option to set				
Option	UDINT	<p>The option to set</p> <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 				
Return parameters	Type	Description				
dRC	DINT	<table border="1"> <tr> <td>0</td><td>Success</td></tr> <tr> <td><0</td><td>Negative error code</td></tr> </table>	0	Success	<0	Negative error code
0	Success					
<0	Negative error code					

If parameter pValue is used, the parameter udValue is then ignored.

See chapter “server configuration” for a list of all options and their possible values.

All values within quotation marks (e.g. “ON”, “OFF”, the default root directory) can be used with parameter pValue; all others must be used with parameter udValue to set the specified option.

If parameter udValue is used, the parameter pValue must be a NIL pointer.

Example

```
dRC : DINT;

// set option ANONYMOUS (anonymous FTP) on

// use parameter pValue
dRC := ocFTPSvr.SetConfig("ON", 0, FTPS_OPT_ANONYMOUS);

// use parameter udValue
dRC := ocFTPSvr.SetConfig(NIL, 1, FTPS_OPT_ANONYMOUS);

// set default root directory
dRC := ocFTPSvr.SetConfig("C:\FTPROOT", 0, FTPS_OPT_DEF_ROOT_DIR);
```

7.6.1.15 UpdateFTPSini

This instruction writes all FTP users and their expanded access rights to the `ftps.ini` file.

```
FUNCTION __CDECL VIRTUAL GLOBAL UpdateFTPSini
VAR_INPUT
    non_blocking : UDINT;
END_VAR
VAR_OUTPUT
    dRC (EAX) : DINT;
END_VAR;
```

Transfer parameters	Type	Description
non_blocking	UDINT	0 blocking mode 1 non-blocking mode
		If non-blocking mode is set, an internal operating system task writes the data to <code>ftps.ini</code> file
Return parameters	Type	Description
dRC	DINT	0 Success

		<0 Negative error code
--	--	------------------------

With this method, data is written to a file. The amount of time the method requires depends on the size of data that must be written to the file. The more FTP users that are available, the more time that is needed to save them to the `ftps.ini` file. The method blocks all other instructions until all data is written to the disk.

It is possible to call the method in non-blocking mode, in which case, an internal operating system task writes the data. The [GetUpdateFTPSiniStatus\(\)](#) method can be used to retrieve the status of the write process in the non-blocking mode.

If more FTP users and/or access rights should be added, changed or removed, this method should be called after all other FTP users have been added, changed or removed.

Example

```
// add FTP user 1, 2, 3,..., remove another, edit another
// add access..., ...

dRC : DINT;

dRC := ocFTPSvr.UpdateFTPSini(0);
if dRC = 0 then
    // FTP users and access rights written to ftps.ini
else
    // error occurred
end_if;
```

7.6.2 Quick Review

The amount of time required by the OS Interface library methods of the `_FTPServer` class is not defined and therefore unpredictable. Calling this function from cyclic or real-time tasks is not recommended.

7.6.3 FTPS.INI

If called with correct parameter settings, all FTP users and access rights added using the command line interface and OS Interface library methods of the `_FTPServer` class are saved to the `ftps.ini` file.

The `ftps.ini` file is a text file and can be edited using any text editor.

Format of the content of the `FTPS.INI` file

U	user name
---	-----------

P	encoded password
GA	general access
R	root directory
A	enlarged access right, path access, path2 access2

Example

```
[U=FTPUser]
P=X@$
GA=R
R=C:\TEMP\
A=C:\TEMP|W
A=D:\|N
```

... next user ...



After the last user entry in the `ftps.ini` file, a newline and carriage return (`\r\n`) is required, otherwise the file cannot be read from the Lasal OS FTP-Server.

The `FTPS.INI` file is stored in the root directory `C:\` (`C:\ftps.ini`)

If the `ftps.ini` file is transferred to the IPC or CPU, the FTP users contained in the file are loaded with the next FTP server start-up.

The password in the file is an encoded password. To encode a password, use the `PwdGen.exe` command. If the content of the file is not valid for a specified user, it will not be saved and added to the registered users at start-up. If the complete content of the file is invalid, no FTP users are loaded.

7.6.4 Error, Warning and Info Logging

With the Command Line Interface (CLI) instruction, `SET DBGLEVEL FTSPSERVER`, information logging can be activated.

`DBGLEVEL FTSPSERVER`

0	no logging
1	error messages only
2	error messages and debug messages
3	extended information

when the default error logging is 1 error messages only are sent.

Error logging can be activated at startup via the autoexec.lsl or manually through the Command Line Interface (CLI) command.

Example

```
SET DBGLEVEL FTPSERVER 2
```

For error logging and debug messages

The FTP server always generates the following extended information entry at startup:

```
*timestamp*;FTPSVR;3;Initializing FTP-Server interface *dll version*
```

7.7 Appendix

7.7.1 Types

Types for Access Rights

Access	Value	DLL Ver.	Description
FTP_ACCESS_NONE	0	01.01.001	No access
FTP_ACCESS_READ	1	01.01.001	Read-only access
FTP_ACCESS_WRITE	2	01.01.001	Full access (read/write/delete)

Types for Reference

Reference	Value	DLL Ver.	Description
FTPS_USR_REF_OS	1	01.01.001	Entry was added by the FTP-Server
FTPS_USR_REF_APPL	3	01.01.001	Entry was added with method AddUser and a value of 0 for the parameter Save

Types for Status

Status	Value	DLL Ver.	Description
FTPS_USR_STAT_ADDED	0x00000001	01.01.001	Entry added, not stored to ftps.ini
FTPS_USR_STAT_CHANGED	0x00000002	01.01.001	Entry changed

FTPS_USR_STAT_STORED	0x000000 04	01.01.001	Entry stored in the ftps.ini file
----------------------	----------------	-----------	-----------------------------------

Types for Configuration Options

Option	Value	DLL Ver.	Description
FTPS_OPT_READ_TMO	2	01.01.001	Read timeout (receive)
FTPS_OPT_WRITE_TMO	3	01.01.001	Write timeout (send)
FTPS_OPT_CMD_TMO	4	01.01.001	Command timeout (receive)
FTPS_OPT_NO_DRIVE	5	01.01.001	No drive on PWD
FTPS_OPT_BACKSLASH	6	01.01.001	Convert to backslash (receive)
FTPS_OPT_FORWARDSLASH	7	01.01.001	Convert to forward slash (send)
FTPS_OPT_MAX_SESSIONS	8	01.01.001	Number of max. logged in FTP clients
FTPS_OPT_DEF_ROOT_DIR	9	01.01.001	Default root directory of the FTP-Server
FTPS_OPT_PASV_PORT	10	01.01.001	Port number of passive data connection
FTPS_OPT_ANONYMOUS	11	01.01.001	Anonymous FTP
FTPS_OPT_ANONYMACCESS	12	01.01.001	Anonymous FTP access
FTPS_OPT_NO_FTPSINI	13	01.01.001	No ftps.ini file
FTPS_OPT_SVR_D_PRIO	14	01.01.001	Priority of the FTP-Server daemon task
FTPS_OPT_SVR_T_PRIO	15	01.01.001	Priority of each connection (FTP client)
FTPS_OPT_WINCOMP_LISTFORMAT	16	01.01.004	Windows compatible list format

Types for Extended User Information

UserInfo	Value	DLL Ver.	Description
FTPS_USR_INFO_REFERENCE	1	01.01.001	Reference of the entry, see Types for Reference
FTPS_USR_INFO_STATUS	2	01.01.001	Status of an entry, see Types for Status
FTPS_USR_INFO_LOGGEDIN	3	01.01.001	Number of logged in FTP users

Other Types

Type	Value	DLL Ver.	Description
FTPSVR_INF	0xFFFF	01.01.001	Value for timeout (infinite)
FTPSVR_USERNAME_LEN	63	01.01.001	Max. user name length
FTPSVR_PASSWORD_LEN	63	01.01.001	Max. password length

7.7.2 Error Values

Error	Value	DLL Ver.	Description
FTPSVR_ERR_VF	-1000	01.01.001	Error initializing internal virtual file system
FTPSVR_ERR_SEMA	-1001	01.01.001	Error creating semaphore
FTPSVR_ERR_DEAMON	-1002	01.01.001	Error creating FTP-Server daemon thread
FTPSVR_ERR_POINTER	-1003	01.01.001	error invalid pointer, NIL
FTPSVR_ERR_PARAM	-1004	01.01.001	error invalid parameter (username too long, ...)
FTPSVR_ERR_NOMEM	-1005	01.01.001	Not enough memory
FTPSVR_ERR_NOT_FOUND	-1006	01.01.001	Entry not found (user, access, ...)
FTPSVR_ERR_EXISTS	-1007	01.01.001	Entry already exists (user, access, ...)
FTPSVR_ERR_INI_OPEN	-1008	01.01.001	Error creating/opening ftps.ini file
FTPSVR_ERR_INI_WRITE	-1009	01.01.001	Error writing to ftps.ini file
FTPSVR_ERR_INI_READ	-1010	01.01.001	Error reading from ftps.ini file
FTPSVR_ERR_UPDATETASK	-1011	01.01.001	Error update task could not be started
FTPSVR_ERR_UPD_BUSY	-1012	01.01.001	Update task busy
FTPSVR_ERR_UPD_NO_TASK	-1013	01.01.001	Update task not started
FTPSVR_ERR_LOGGED_IN	-1014	01.01.001	User logged in
FTPSVR_ERR_NO_INTERFACE	-1015	01.01.001	Error, no FTP-Server interface available

7.7.3 Flags

Flags for change a FTP user (EditUser)

Flag	Value	DLL Ver.	Description
FTPSVR_CHANGE_PASSWORD	0x00000001	01.01.001	Change password
FTPSVR_CHANGE_GENERALACCESS	0x00000002	01.01.001	Change general access
FTPSVR_CHANGE_ROOTDIR	0x00000004	01.01.001	Change rootdir

8 Logfile

8.1 General

This document explains the entries in the EVENT00.LOG log file. In addition, possible error sources and solutions are described. In this document, the most important entries are listed.

Most log file entries begin with the date and time.

The date format is year/month/day, for example: "08/11/29;".

The time format is hours:minutes:seconds.milliseconds, for example: 13:49:26.347;

Some entries in the log file begin with *I*6E:. The „*I“ indicates the entry was triggered by a hardware interrupt. The following number is a running hexadecimal counter.

Entries at the end of the log file are always the most current.

8.2 A Detailed Breakdown of Individual Log Entries

8.2.1 System

Entry	PowerON,OS:01.02.035(Oct 15 2008,15:22:20),ETV 0811,PLC-SerNbr.01973776,FPGA V14 Or ----- PowerON,OS:01.02.053(Apr 21 2009,10:44:57),CIPC,PLC-SerNbr.01163204,FPGA V23 -----
Description	The system was activated. Operating system version: 01.02.035 release date: 15. October 2008 release time: 15:22:20 CPU type: ETV0811 Serial number: 01973776 FPGA version: 1.4 (FPGA version number is not always provided)
Cause / Solution	The system was activated.
Entry	DLL_LoadLib: Loading library "rexx.dll" failed, rc=2
Description	An error has occurred while loading DLL (here „rexx.dll“). "rc" is the return value of the LoadLibrary function.
Cause / Solution	The file (here "rexx.dll") is missing from the system or in the wrong place and cannot be found.
Entry	runStatus changed from 0 to 26 Or runStatus changed from RUN RAM(0) to WAIT(26)

Description	The control's CPU status has changed. The CPU status has changed from 0 to 26. The numbers must be interpreted with the table for the CPU status.
Cause / Solution	The control program is started or ended. More information can be found in the chapter CPU Status.
Entry	PrjName:Class,Chksum:0xC3492D3D
Description	The application with the name "Class" and the checksum 0xC3492D3D is started. If the checksum has changed since the last entry, a different software version was loaded (update or stage of development).
Cause / Solution	Info
Entry	SFB on (00388264)
Description	ShadowFrameBuffer was activated. From now on, graphic instructions draw in a memory area that is not visible.
Cause / Solution	Info
Entry	SFB off (004225F0)
Description	ShadowFrameBuffer was activated. From now on, graphic instructions draw in a memory area that is visible.
Cause / Solution	Info
Entry	*I*9A:Power fail! Or USV Power down recognized
Description	A voltage disruption was detected.
Cause / Solution	The control is deactivated.
Entry	MakeBootDisk (ONLINE): filename = C:\IPC.RTB MakeBootDisk (ONLINE): result = 0
Description	An operating system update was performed by the user.
Cause / Solution	Info

Entry	Reboot
Description	A restart was executed.
Cause / Solution	Info
Entry	WATCHDOG Error!
Description	The watchdog was not triggered for 130 ms. The periphery (DIAS-Bus / VARAN-Bus / C-DIAS) is in failsafe mode.
Cause / Solution	<p>The real-time task was blocked for at least 30 ms. This mostly occurs in combination with a real-time runtime error.</p> <p>For entering directly after the initialization phase: do not use the DIAS and/or VARAN bus (no real-time runtime error)</p>
Entry	<p>Exception 0xC0000094, EIP=0x00E4D164 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=WSPO_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 00 0E82FDB0:00 00 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..... 0E82FDF0:00 00 00 00 03 18 00 00 FF FF FF FF FF FF FF FF FF 0E82FE00:00 00 7C 01 00 4C 7F 0E 00 00 00 00 94 00 00 C0L..... 0E82FE10:DC FD 82 0E 50 1B 80 0E 00 FE 82 0E F4 FB 82 0EP.....</pre>

```
0E82FE20:FF FF FF FF DC 9B 2C 00 C8 4D 36 00 00 00 00 00 .....M6.....
0E82FE30:C8 C1 89 0E 71 D3 23 00 30 70 89 0E 03 18 00 00 ....q.#.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 0E .....L..
0E82FE60:01 00 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 SSSSSSSSSSSSS
SSSSSSSSSSSSSSSS
0E82FED0:53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 53 SSSSSSSSSSSSS
SSSSSSSSSSSSSS
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     :00100000-001BFF88,001BFD0C,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.stl::_READ@OHMMETER
(000327)
0E82FDD8:EIP=0x0026B348
0E82FE34:EIP=0x0023D371
0E89C1CC:EIP=0x00000000
```

or

```
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.stl::_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 #
.....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 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 :00100000-001BFFF88,001BFD0C,780160d
Idle Task :000CCDAO-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
```

Description	Exception with exception code and instruction pointer (EIP) With the dump, it is possible to search for the application error offline. Exception coded with description: 0xC0000005 : ACCESS EXCEPTION A logical memory address was accessed that doesn't exist. I.e.: memory access of the Null pointer 0xC000001D : ILLEGAL INSTRUCTION An undefined instruction should be executed. 0xC000008C : ARRAY BOUNDS EXCEEDED Access outside of the defined array limits 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	
	A result should be calculated, which cannot be calculated because the actual values generate an invalid result.
0xC0000096 : PRIVILEGED INSTRUCTION	
	A privileged instruction should be executed that is not allowed.
0xC00000FD : STACK_OVERFLOW	
	A Stack has overflowed.

Entry	Real-time RUNTIME Error;Task=WSP0_RT;Object=CALLVARAN1
Description	<p>The total duration of all real-time objects exceed the maximum time.</p> <p>2 ms for 386 CPUs</p> <p>1 ms for all other CPUs</p> <p>An error has occurred in the WSP0_RT task (Real-time Task).</p> <p>An error has occurred in the CALLVARAN1 object.</p> <p>The entry has a dump entry => see Exception</p>
Cause / Solution	<p>Optimize the application's real-time task (RtWork).</p> <p>Reduce the clock time for the real-time task of all objects.</p> <p>Correct application errors</p> <p>CPU is overloaded in real-time => use a higher capacity CPU.</p>

Entry	RUNTIME Error;Task=WSP0_CT;Object=AXLESCHEDULER1
Description	<p>The total duration of all cyclic objects exceed the maximum time.</p> <p>An error has occurred in the WSP0_CT task (cyclic task)</p> <p>An error has occurred in the AXLESCHEDULER1 object.</p> <p>The entry has a dump entry => see Exception</p>
Cause / Solution	<p>Optimize the application task (CyWork).</p> <p>The time can be configured using 2 global system variables:</p> <ul style="list-style-type: none"> Runtime: time remaining

	<ul style="list-style-type: none"> • SWRuntime: pre-selected value for the runtime counter or set using "autoexec.lst" (SET RUNTIME 30) <p>Use higher capacity CPU</p>
Entry	Background RUNTIME Error;Task=WSP0_BT;Object=CLASS01
Description	<p>The total duration of all background objects exceeds the maximum time.</p> <p>An error has occurred in the WSP0_BT task (background).</p> <p>An error has occurred in the CLASS01 object.</p> <p>The entry has a dump entry => see Exception</p>
Cause / Solution	<p>Optimize the application's background task (background)</p> <p>The time can be configured using 2 global system variables:</p> <ul style="list-style-type: none"> • BTRuntime: time remaining • SWBTRuntime: pre-selected value for the runtime counter <p>Use higher capacity CPU</p>
Entry	*** overflow at 08/05/08;09:51:09.214; ***
Description	Too many log entries at once. One or more log entries are lost.
Cause / Solution	Info
Entry	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
Description	Fatal system error
Cause / Solution	Correct application program errors.

8.2.2 USB

Entry	USB;2;USB connect VID=0000 PID=0000 USB;2; Class=9 SubClass=0 Protocol=0
Description	A USB device was detected and connected with VendorID 0000 and ProductID 0000 Class=9 SubClass=0 Protocol=0
Cause / Solution	A USB device was found and connected Note: internal control devices are also listed here.

Entry	USB;2;USB disconnect VID=046d PID=c30e
Description	A USB device was removed.
Cause / Solution	USB stick or keyboard was disconnected.

8.2.3 DIAS

Entry	DIASMaster: PlTime = 4
Description	The DIAS bus clock time is 4 ms.
Cause / Solution	Info
Entry	DIASMaster: DIASType = 2, CdiasType = 2
Description	DIAS bus type: 0 = None 1 = Standard Master 2 = Intelligent Master C-DIAS bus type: 0 = None 1 = Memory mapped 2 = Memory mapped
Cause / Solution	Info
Entry	DIASMaster: PLL is locked and the intelligent master's Rt program starts (1)
Description	The hardware class is synchronized with the hardware and the intelligent Master is now started.
Cause / Solution	Info
Entry	DIASMaster: We are Sync -> Rt program of intelligent master starts again (Sync Module Place: 0) (2)
Description	Hardware class synchronized. The answer from DIAS bus Sync instruction is installed in the module number with station number 0 With 'Sync Module Place: 255', no module was found on the DIAS bus that uses the DIAS bus Sync instruction.
Cause / Solution	Info
Entry	DIASMaster: MoveStartPointer -> Start moment of IM - program is set (Rt - Startpoint: 3900us, AddTime: 0ns) (3)
Description	Start point of the intelligent Master was set to 3900 μ s,

	For each CIC with a repeater (CIC012/121/212/221) additional time is needed to process the signal. This time is assigned with AddTime.
Cause / Solution	Info
Entry	DIASMaster: One Time point was read -> start updating real-time tasks now (0 Rt - DIAS accesses) (4)
Description	The time until the next scan of the intelligent Master was read and starting from this time point, the real-time hardware classes are processed, whereby the number of times the DIAS bus is accessed is assigned (in this case = 0).
Cause / Solution	Info
Entry	DIASMaster: Init ends
Description	The DIAS bus initialization is complete.
Cause / Solution	Info
Entry	DIASError on DIV511 (DeviceAddress: 10000, DIASModulePlace: 9)
Description	A DIAS error on the DIV511 DIAS bus with the device address hex 10000 as occurred at station number 9. With DIAS error at station 63, the DIAS bus Sync instruction was not answered.
Cause / Solution	If the station number is not assigned in the network, the DIAS error can also be a hardware class error. It can occur during the system shutdown. Hardware problems: See chapter Troubleshooting DIAS Bus Problems
Entry	DIASMaster: Unable to create flag for asynchronous access!
Description	No flag could be assigned for the OS call.
Cause / Solution	No further flags possible or several DIASMasterC objects are placed.
Entry	DIASMaster: Wrong DIASMaster Time for new PLL! SystemPeriod: %d, DIASMasterTime: %d
Description	The DIASMasterTime must be a multiple of (SystemPeriod / 100).
Cause / Solution	Set Real-time setting for the DIASMaster object to a multiple of the system period time (Default: 1ms)
Entry	Watchdog Error (RTSSW<>4) -> DIASMasterOn = 0 and ImOff = 1

Description	The DIAS/C-DIAS Watchdog was not triggered because the DIASMaster object has not been called for more than 30 ms. è Hardware-Reset am DIAS- and C-DIAS Bus. This message is given, when the initialization of the DIASMaster object is not yet completed.
Cause / Solution	Correct programming error Possible infinite loop in an Init method
Entry	Watchdog Error -> DIASMasterOn = 0 and lmOff = 1
Description	The DIAS/C-DIAS Watchdog was not triggered because the DIASMaster object has not been called for more than 30 ms. -> Hardware-Reset on DIAS- and C-DIAS Bus. This message is given, when the initialization of the DIASMaster object has been completed.
Cause / Solution	Correct programming error Possible infinite loop in a real-time task.
Entry	DIASMaster: Watchdog on C-DIAS and DIAS - Bus was triggered! (The RtTask of DIASMasterC - Class was set Off for more than 30ms => Hardware reset on DIAS and CDIAS - Bus)
Description	The watchdog was not triggered for 130 ms. Die CDIAS / DIAS-Bus is changed to Fail Safe mode.
Cause / Solution	The real-time task was blocked for at least 30 ms. This mostly occurs in combination with a real-time runtime error.
Entry	DIASMaster: Timeout on asynchron methods! => DIASBus is set off!!
Description	With the asynchronous DIAS bus instructions in the intelligent Master, a time out has occurred.
Cause / Solution	The intelligent master is deactivated (possibly due to a Watchdog error). Sequence errors.
Entry	DIASMaster: But no Watchdog on CDIAS and DIAS-Bus
Description	Status of the Watchdog to the previous log entry.
Cause / Solution	Info
Entry	DIASMaster: RISC is set to -> DIASMasterOn = 0 and lmOff = 1
Description	Intelligent Master was deactivated.
Cause / Solution	Sequential errors / Info
Entry	DIASMaster: Rt of IM starts while Rt of DIASMaster is running!
Description	Real-time program of the intelligent master starts while the hardware classes are still running.

Cause / Solution	Time problem in the application's real-time program. Optimize the application's RtWork. Install higher capacity CPU.
Entry	DIASMaster: Timeslice overrun!
Description	Timeslice error in the intelligent master
Cause / Solution	Time problem in the application's real-time program. Optimize the application's RtWork. Install higher capacity CPU.
Entry	DIASMaster: Too many cyclic calls
Description	The intelligent master DPRAM for cyclic objects is full.
Cause / Solution	Set more hardware classes to real time.
Entry	DIASMaster: Too many real-time calls
Description	The intelligent master DPRAM for real-time objects is full.
Cause / Solution	Set more hardware classes to cyclic.
Entry	DIASMaster: Cyclic program of intelligent master is too large
Description	The intelligent master DPRAM for cyclic objects is full.
Cause / Solution	Set more hardware classes to real time.
Entry	DIASMaster: Real-time program of intelligent master is too large
Description	The intelligent master DPRAM for real-time objects is full.
Cause / Solution	Set more hardware classes to cyclic.
Entry	DIASMaster: Intelligent master is setting Off
Description	Intelligent master has been switched-off because of the previous log entry.
Cause / Solution	Info
Entry	DIASMaster: Timeout while synchronization

Description	Problems at hardware synchronization.
Cause / Solution	Real-time is off => set to i.e. 1 ms
Entry	DIAS Error (9)
Description	DIAS-Error at station number 9 (Number is in Hex => number set on the Hex switch) With a DIAS error at station 3F, the DIAS bus Sync instruction was not answered. The module that did not answer can be seen in the log entry "DIASMaster: We are Sync -> Rt program of intelligent master starts again (Sync Module Place: 0) (2)" (in this case 0, Caution: this number is in decimal).
Cause / Solution	If the station number is not assigned in the network, the DIAS error can also be a hardware class error. It can occur during the system shutdown. Hardware problems: See chapter Troubleshooting DIAS Bus Problems
Entry	DIAS Risc Error (6)
Description	Software operating error in the intelligent master The number is not relevant.
Cause / Solution	Contact SIGMATEK

8.2.4 S-DIAS

Entry	SDIAS_LOCAL;2;PCI config interface not found!
Description	No PCI interface is available and PCI devices are also not supported.
Cause / Solution	Info
Entry	SDIAS_LOCAL;2;SDIAS_MANAGER_CFG Space in FPGA not found!
Description	No S-DIAS Manager is available in the FPGA.
Cause / Solution	No local S-DIAS bus available. The S-DIAS clients can only be used behind the bus coupler module (like VI021, for example).
Entry	SDIAS_LOCAL;2;VARAN_PLL_REGISTER Space in FPGA not found!
Description	The nanoseconds PLL needed for the local S-DIAS Manager is now available.
Cause / Solution	The local S-DIAS Manager cannot be used, since the FPGA is inconsistent.

Entry	SDIAS_LOCAL;2;Found ctrl=%p, dpram=%p, pll=%p
Description	Listing of the virtual start address for FPGA components required for the S-DIAS Manager.
Cause / Solution	Info
Entry	SDIAS_LOCAL;2;length bigger than area
Description	When copying the isochronous data from the shadow buffer into the DPRAM or reverse, an error occurred because the length of the source data exceeded the size of the destination buffer.
Cause / Solution	<p>The length of the data from the data object exceeds size of the available DPRAM or shadow buffer.</p> <p>The number of bytes in the isochronous data objects must be reduced. This can be performed by either changing the topology (e.g. removing unnecessary S-DIAS modules, placing S-DIAS modules from the local S-DIAS bus onto the S-DIAS bus behind the bus coupler module) or optimizing the S-DIAS memory access.</p>
Entry	SDIAS_LOCAL;3;No SDIAS bus available
Description	There is no S-DIAS Manager available for the local S-DIAS bus.
Cause / Solution	<p>No S-DIAS Manager could be initialized for the local S-DIAS bus, since either some required hardware components were not recognized by the system or this error occurred during the initialization.</p> <p>This log entry indicates that the CPU does not support local S-DIAS buses or if this is not the case, there is a hardware defect.</p>
Entry	SDIAS_LOCAL;2;SDIAS Manager task creation failed!
Description	The task for processing data in the local S-DIAS Manager could not be generated.
Cause / Solution	<p>Due to insufficient resources, the task for the S-DIAS Manager could not be assigned and started.</p> <p>The size of the OS heap is set too low and must be expanded accordingly.</p>
Entry	SDIAS_LOCAL;2;SDIAS subsystem running...
Description	The S-DIAS Manager for the local S-DIAS bus could be successfully initialized and the corresponding process started.
Cause / Solution	Info
Entry	SDIAS_LOCAL;2;SDO request failed with global error: channel %d, client: %u, command: 0x%04X, error code: 0x%08X
Description	Access to an S-DIAS client has failed.

Cause / Solution	<p>For a more detailed analysis of the failed access, the following information is listed in the log entry.</p> <p>Communication channel number (SDO channel)</p> <p>Slot of the affected S-DIAS client</p> <p>Failed command</p> <p>Error code</p>
Entry	SDIAS_LOCAL;2;Mutex creation failed
Description	A mutex required for the SDO communication with S-DIAS participants could not be created.
Cause / Solution	Due to insufficient resources, the mutex could not be assigned or created.
	The size of the OS heap is set too low and must be expanded accordingly.
Entry	SDIAS_LOCAL;1;SDIAS_Manager_API_Init: SDIAS SDO task create failed
Description	The task for SDO communication with S-DIAS participants could not be created.
Cause / Solution	Due to insufficient resources, the task for SDO communication could not be assigned and started.
	The size of the OS heap is set too low and must be expanded accordingly.
Entry	SDIAS_LOCAL;2;SDIAS_Manager_Task_Execute: error
Description	An error has occurred while running an asynchronous task list in the S-DIAS Manager.
Cause / Solution	Info
Entry	SDIAS_LOCAL;2;SDIAS_Module_CPLD_Com_Write_Data: request outside of SDIAS manager control detected (%lu)
Description	Write access to the S-DIAS Manager memory failed.
Cause / Solution	An attempt was made to access an invalid S-DIAS Manager. To identify the failed access, the start address of the write access is output in the log entry.
Entry	SDIAS_LOCAL;2;SDIAS_Module_CPLD_Com_Read_Data: request outside of SDIAS manager control detected (%lu)
Description	Read access to the S-DIAS Manager memory failed.
Cause / Solution	An attempt was made to access an invalid S-DIAS Manager. To identify the failed access, the start address of the read access is output in the Log entry.
Entry	SDIAS_LOCAL;2;SDIASManager: iHWTAddModule2Node failed with %d for device %d
Description	An S-DIAS client could not be added to the hardware tree.

Cause / Solution	The cause for this is contained in the log message as return code, as well as the slot of the affected S-DIAS client.
Entry	SDIAS_LOCAL;2;----> node %p, rootnode %p, pVNode %p, pSdiasManager %p
Description	An S-DIAS client was added to the hardware tree. The Log entry contains the address of the addition and the addresses of the overlying module nodes.
Cause / Solution	Info
Entry	SDIAS_LOCAL;2;SAFETY_LVDS: Add module (0x%x/0x%x) to the SafetyCPU BUSINTERNAL tree
Description	An S-DIAS Safety CPU expansion card for bus type 0x92 to the hardware tree. The device and sub-device ID of the added expansion card is contained in the log entry.
Cause / Solution	Info
Entry	SDIAS_LOCAL;2;SAFETY_LVDS: Add module (0x%x/0x%x) to the SafetyCPU CANBUS tree
Description	A CAN bus participant of the S-DIAS Safety CPU was added in the hardware tree for bus type 0x87. The device and sub-device ID of the added CAN bus participant is contained in the Log entry.
Cause / Solution	Info
Entry	SDIAS_LOCAL;2;SAFETY_LVDS: Found a module (0x%x/0x%x) with unknown bus topology! Node will be ignored.
Description	An S-DIAS Safety CPU expansion module with a non-supported bus type was detected. The device and sub-device ID of the added expansion module is contained in the log entry.
Cause / Solution	The detected bus type was not recognized by the operating system. This can be for example, that the OS version used does not support the detected bus type (in such a case, the operating system is updated) or the Safety CPU hardware is defective.
Entry	SDIAS_LOCAL;2;SDIAS_Module_Obj_Config_Memory_Get failed for SAFETY CPU
Description	Reading the identification of the expansion cards in an S-DIAS CPU has failed.
Cause / Solution	When accessing memory in the S-DIAS CPU, an error has occurred. This error can result from a number of causes, for example, defective hardware, bad contacts or disruptions.
Entry	(BusInterfaceVARAN::Init) GetLongDOsPossible was called with the wrong VARANManagerNr.

Description	In the VARAN Manager, a non-zero VARAN Manager number was entered. The availability of LongDOs cannot be determined.
Cause / Solution	Info
Entry	(BusInterfaceVARAN::AddDO) The FPGA of the VARAN Manager does not support Long DOs. Newer Version required!
Description	The VARAN Manager still does not support "LongDOs" (data objects > 128 bytes).
Cause / Solution	Use newer FPGA version for the respective platform.
Entry	(BusInterfaceVARAN::BusInterfaceVARAN) Failed to get VARAN-CIL (interface to OS)!
Description	Operating interface for VARAN could not be found.
Cause / Solution	User newer OS version.
Entry	(BusInterfaceSDIASInternal::SetSyncData) CycleTime of SDIAS Manager has to be a multiple of the maintimer setting!
Description	The S-DIAS bus time setting is not a multiple of the CPU main timer.
Cause / Solution	Adjust main timer or S-DIAS bus time.
Entry	(BusInterfaceSDIASInternal::SetSyncData) HwControl realtime setting has to be a multiple of the maintimer setting!
Description	The realtime setting is not a multiple of the CPU main timer.
Cause / Solution	Adjust main timer or realtime setting.
Entry	(BusInterfaceSDIASInternal::SetSyncData) Invalid constellation of MainTimer and cycle time of SDIAS! CycleTime/MainTimer has to be smaller than 256!
Description	The S-DIAS bus time setting is too high for the selected CPU main timer.
Cause / Solution	S-DIAS bus time [μs] / main timer[μs] bust be less than 256.
Entry	(SdiasManager::SdiasManager) Failed to get SDIAS OS-Interface. A newer OS is necessary to use SDIAS
Description	The operating system interface for S-DIAS could not be retrieved.

Cause / Solution	The version of the operating system used does not support the S-DIAS bus. Use newer operating system version.
Entry	(BusInterfaceSDIASInternal::HwControlLogin) CycleTime of SDIAS Manager has to be a multiple of the HwControl realtime setting!
Description	The S-DIAS bus time setting is not a multiple of the hardware control object.
Cause / Solution	Adjust the S-DIAS bus time or realtime.
Entry	(BusInterfaceSDIASInternal::HwControlLogin) Response of login at HwControl has an invalid size
Description	The version of the class is not compatible with the HWControl used.
Cause / Solution	Update hardware classes.
Entry	(BusInterfaceSDIASInternal::HwControlLogin) Failed to install callbacks at HwControl
Description	The version of the HwControl used is too old.
Cause / Solution	Update HwControl
Entry	(SdiasManager::Init) SDIAS Manager object inactive");
Description	The S-DIAS Manager does not run or is disabled.
Cause / Solution	With a local S-DIAS bus: subsequent errors occur when the operating for the S-DIAS bus was not found. à Update operating system. S-DIAS bus via VARAN (e.g.: VI021) Either the VARAN Manager is deactivated due to an error or the VI021 is set to transparent.
Entry	(SdiasManager::SdiasManager) Failed to get SDIAS OS-Interface. A newer OS is necessary to use SDIAS";
Description	The operating system interface for S-DIAS could not be retrieved.
Cause / Solution	The version of the operating system used does not support the S-DIAS bus Use newer operating system version.
Entry	(SdiasManager::Init) Failed to get bus cycle time via BusInterface");
Description	The bus cycle time could not be determined.
Cause / Solution	Subsequent errors occur when the bus and real time settings are not possible. See further log entries for more information.

Entry	(SdiasManager::Init) Failed to add DO for the asynchronous read data");
Description	A data object could not be created.
Cause / Solution	See further log entries for more information.
Entry	(SdiasManager::Init) Failed to add DO for the asynchronous write data");
Description	A data object could not be created.
Cause / Solution	See further log entries for more information.
Entry	(SdiasManager::CyWork) No SDIAS Client objects projected/connected to the SDIAS Manager! Unable to continue with Initialization!");
Description	No S-DIAS client objects were found/connected in the software.
Cause / Solution	For information only. The initialization or the S-DIAS bus is not continued, the remaining hardware is not affected.
Entry	(SdiasManager::RtWork) Error at toggle bit of iso write task of SDIAS");
Description	The toggle bit of the isochronous write task has the wrong condition.
Cause / Solution	The write task was not completely executed. Check the bus time setting and ISOStartPoints.
Entry	(SdiasManager::RtWork) Error at toggle bit of iso read task of SDIAS");
Description	The toggle bit of the isochronous read task has the wrong condition.
Cause / Solution	The read task was not completely executed. Check the bus time setting and ISOStartPoints.
Entry	(SdiasManager::RtWork) Error at iso write task of SDIAS");
Description	The isochronous write task has detected an error.
Cause / Solution	An error has occurred during write access. <ul style="list-style-type: none"> • Bus connection is bad / disrupted • Supply voltage failed or break
Entry	(SdiasManager::RtWork) Iso write task of SDIAS hasn't been completed");
Description	The isochronous write task was not completed in the last cycle.
Cause / Solution	Insufficient time allotted for the write task. Check the bus time setting and ISOStartPoints.

Entry	(SdiasManager::RtWork) Iso read task of SDIAS hasn't been completed! Check setting of IsoStartPoint and cycle time!');
Description	The isochronous read task was not completed in the last cycle.
Cause / Solution	Insufficient time allotted for the read task. Check the bus time setting and ISOStartPoints.
Entry	(SdiasManager::Init) Missing SDIAS Place 0x{0} in Configuration!
Description	A hardware module was found for which no hardware class is placed.
Cause / Solution	INFORMATION A hardware slot [hex] containing a module was found, but no hardware class is available.
Entry	(SdiasManager::CyWork) Wrong device ID at SDIAS Place 0x{0}, i);
Description	The hardware module does not match the software.
Cause / Solution	INFORMATION The hardware slot [hex] does not match the software.
Entry	(SdiasManager::CyWork) Failed to initialize module on SDIAS Place 0x{0} via InitModule-Interface", ActModule[i]);
Description	An S-DIAS module returns an error during the initialization.
Cause / Solution	INFORMATION See further log entries for more information.
Entry	(SdiasManager::CyWork) Init has been blocked for too long! Initialization of SDIAS Manager timed out! Blocked step: 0x{0}", InitSSW\$UDINT);
Description	An S-DIAS module returns an error during the initialization.
Cause / Solution	INFORMATION See further log entries for more information.
Entry	(SdiasManager::AddAccess) Invalid Place number: 0x{0}. Maximum is 0x{1}", Place, SDIAS_MAX_PLACE_NR);
Description	An invalid S-DIAS Place number was assigned.
Cause / Solution	Check Place settings in the software.
Entry	(SdiasManager::SetRequiredError) Required Error on SDIAS Place 0x{0}", Place);
Description	A Required error was triggered for a missing module.

Cause / Solution	<ul style="list-style-type: none"> Check whether all modules in the hardware configuration are connected and powered. Check bus connection. Check voltage supply
------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

8.2.5 VARAN

Entry	MAC-Address not available, VARAN to Ethernet Bridge (0)!														
Description	<p>No MAC address is defined for the VARAN manager.</p> <p>The MAC address for the network card is used.</p>														
Cause / Solution	Info														
Entry	Task VARAN Task is terminating!														
Description	The task (here "VARAN Task") was ended because it triggered an exception.														
Cause / Solution	<p>An error in the operating system or callback function in the hardware class.</p> <p>=> Update the operating system and hardware class.</p>														
Entry	<p>VARAN Callback (DEA30)(5)</p> <p>OR</p> <p>VARAN Callback (TIME SLICE ERROR IRQ: VMC052,ADDR:0x0,Port:0)</p>														
Description	<p>The application was notified of a status change on the VARAN bus.</p> <p>The last number in the old variant is the status code.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;">0</td><td>Device has been connected (CONNECT or CONNECT PREV)</td></tr> <tr><td>1</td><td>Device has been removed (DISCONNECT)</td></tr> <tr><td>2</td><td>Access error in a required module (required-frame error)</td></tr> <tr><td>3</td><td>Access error in a non-required module (not-required-frame error)</td></tr> <tr><td>4</td><td>Fatal system error (FATAL ERROR)</td></tr> <tr><td>5</td><td>Timeslice error in the application (TIME SLICE ERROR IRQ)</td></tr> <tr><td>6</td><td>Watchdog (WATCHDOG ERROR)</td></tr> </table> <p>In this new variant, the status code is in clear text + the device in which the error occurred.</p>	0	Device has been connected (CONNECT or CONNECT PREV)	1	Device has been removed (DISCONNECT)	2	Access error in a required module (required-frame error)	3	Access error in a non-required module (not-required-frame error)	4	Fatal system error (FATAL ERROR)	5	Timeslice error in the application (TIME SLICE ERROR IRQ)	6	Watchdog (WATCHDOG ERROR)
0	Device has been connected (CONNECT or CONNECT PREV)														
1	Device has been removed (DISCONNECT)														
2	Access error in a required module (required-frame error)														
3	Access error in a non-required module (not-required-frame error)														
4	Fatal system error (FATAL ERROR)														
5	Timeslice error in the application (TIME SLICE ERROR IRQ)														
6	Watchdog (WATCHDOG ERROR)														
Cause / Solution	<p>Statuscode.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 10%;">0,1</td><td>Info</td></tr> </table>	0,1	Info												
0,1	Info														

- 2.3 Hardware problems: See chapter [Troubleshooting VARAN Bus Problems](#)
 This can occur during the system shutdown or hardware problems: see chapter [Troubleshooting VARAN Bus Problems](#).
- 5.6 Time problem in the application's real-time program.
 Optimize the application's RtWork.
 Install higher capacity CPU
 Set a slower clock time for all RtWork.
 It can occur during the system shutdown.
 This can occur while ending the application (Reset, Runtime, Division by 0,)

Entry	VARAN Watchdog Error
Description	The watchdog was not triggered for 130 ms. The VARAN bus switches to the fail-safe mode.
Cause / Solution	The real-time task was blocked for at least 30 ms. This mostly occurs in combination with a real-time runtime error. Time problem in the application's real-time program. Optimize the application's RtWork. Install higher capacity CPU.
Entry	VARANManager 0: Started with Version 1.16
Description	The VARAN Manager class starts with version 1.16
Cause / Solution	Info
Entry	VARANManager 0: VARANManagerTime (%d) is not a multiple of MainTimer (%d)!
Description	The VARAN bus cycle time must be a multiple of the Main timer (clock signal for the operating system and the fastest possible cycle time).
Cause / Solution	Set the bus cycle time to a multiple of the OS main timer or set the main time accordingly. The default for the main timer and the VARAN cycle time is 1 ms.
Entry	VARANManager 0: Client - Function not available but activated! (at least OS - Version 1.1.217 required)
Description	The VARAN manager client function is active but not available with this operating system version.
Cause / Solution	Update operating system to 1.1.217 or higher.
Entry	VARANManager 0: VARANManagerTime (%d) is not a multiple of System Period Time (%d)!
Description	The VARAN bus cycle time must be a multiple of the system period (= clock time of the RANManager FPGA)

Cause / Solution	Set the VARAN bus cycle time to a multiple of the FPGA system period.
Entry	VARANManager 0: Unable to create semaphore for async access!
Description	No flag could be assigned for direct access.
Cause / Solution	No further flags possible or several VARANManager objects are placed.
Entry	VARANManager 0: IRQ Task Time not a multiple of VARANManager Time!
Description	The VARAN Manager must be a multiple of the interrupt task time!
Cause / Solution	Correct the IRQ Task time.
Entry	VARANManager 0: No Memory for VARAN Manager available! (Make Rt - Handles: 50:
Description	No memory could be allocated for the real-time handles (all UpdateRt methods of the VARAN hardware classes are called).
Cause / Solution	Reserve less memory in the application
Entry	VARANManager 0: No Memory for VARAN Manager available! (Make Cy - Handles: 50:
Description	No memory could be allocated for the cyclic handles (all UpdateCy methods of the VARAN hardware classes are called).
Cause / Solution	Reserve less memory in the application
Entry	VARANManager 0: No Memory for VARAN Manager available! (Extend Rt - Handles: 50:
Description	The memory for the real-time handles (all UpdateRt methods of the VARAN hardware classes are called) could not be extended.
Cause / Solution	Reserve less memory in the application.
Entry	VARANManager 0: No Memory for VARAN Manager available! (Extend Cy - Handles: 50:
Description	The memory for the cyclic handles (all UpdateCy methods of the VARAN hardware classes are called) could not be extended.
Cause / Solution	Reserve less memory in the application
Entry	VARANManager 0: Error happened with ErrorCode %d!

Description	Error has occurred in the VARANManager ErrorCode:
	-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

	<ul style="list-style-type: none"> -28 VARANMANAGER_SPI_FLASH_NO_ACCESS -29 VARANMANAGER_CLIENT_NOT_READY -30 VARANMANAGER_CLIENT_DISABLED -31 VARANMANAGER_CLIENT_CANT_ENABLE
Cause / Solution	Info
Entry	VARANManager 0: Is set off because of last Error!
Description	The Manager stopped by the last error.
Cause / Solution	Info
Entry	VARANManager 0: Timeslice Error! Manager is set to off!
Description	Real-time program in the VARAN Manager is not yet complete when the CPU reaches its real-time task.
Cause / Solution	<ul style="list-style-type: none"> Set ISOStartPoint in the VARAN Manager correctly. Reduce the clock speed.
Entry	VARANManager 0: Timeslice Error Interrupt, Manager is set off!
Description	The real-time task in the hardware classes is not yet finished with processing the DPRAM data when the VARAN Manager wants to start its real-time task.
Cause / Solution	<ul style="list-style-type: none"> Set ISOStartPoint in the VARAN Manager correctly. Reduce the clock speed. Optimize the application's RtWork.
Entry	VARANManager 0: Watchdog Error Interrupt, Manager is set to off!
Description	The watchdog was not triggered for 130 ms. The VARAN bus switches to the fail-safe mode.
Cause / Solution	<ul style="list-style-type: none"> The real-time task was blocked for at least 30 ms. This mostly occurs in combination with a real-time runtime error. Time problem in the application's real-time program. Optimize the application's RtWork. Install higher capacity CPU
Entry	VARANManager 0: Required Module not ready, Manager is set to off!
Description	A VARAN-Client module that is "required" in the application (= mandatory)

Cause / Solution	Possible wiring error in the power supply or the VARAN bus cable to the corresponding client.
Entry	VARANManager 0: Device message DIASError on DIV511 (DeviceAddress: 10000, DIASModulePlace: 9: VARANManager 0: Error on Device, Manager is set off!!
Description	A DIAS error on the DIV511 DIAS bus with the device address hex 10000 has occurred at station number 9. With a DIAS error at station 63, the DIAS bus Sync instruction was not answered. Change the DIAS and VARAN bus to the Fail Safe/Reset mode.
Cause / Solution	If the station number is not assigned in the network, the DIAS error can also be a hardware class error. It can occur during the system shutdown. Hardware problems: See chapter Troubleshooting DIAS Bus Problems
Entry	VARANManager 0: SuperiorSystemTime is 0!
Description	If a VMC insert card is used, the system time from the primary system must be entered in the SuperiorSystemTime client of the VARANManager, so that both systems can synchronize.
Cause / Solution	Enter the primary system time in the SuperiorSystemTime client of the VARANManager.
Entry	VARANManager 0: Couldn't find sync out for switching alternating buffer
Description	When using VMC: The FPGA components to enable the alternating buffer switch could not be found.
Cause / Solution	Either the hardware is defective or the operating system has found a false VMC interface.
Entry	VARANManager 0: VARANTime of the primary and secondary systems have to be multiples of each other
Description	When using the VMC: the VARAN cycle time in the primary system must be a multiple of the VARAN cycle time of the secondary system or vice versa, otherwise synchronization is not possible.
Cause / Solution	Adjust the cycle time of the primary or secondary system.
Entry	VARANManager 0: VARANTime of the primary system and the maintimer of secondary system must be multiples of each other.
Description	When using the VMC: the VARAN cycle time in the primary system must be a multiple of the OS clock time (Maintimer) of the secondary or vice versa, otherwise synchronization is not possible.

Cause / Solution	Adjust the VARAN cycle time of the primary system or the Maintimer of the secondary system accordingly.
Entry	VARAN required-frame error ECE06800 OR VARAN required-frame error DO:ECE06800
Description	An access error has occurred in a required module. The number shows the pointer to the data object with the error. This entry is always in combination with other entries, with which the defective module can be found.
Cause / Solution	It can occur during the system shutdown. Hardware problems: See chapter Troubleshooting VARAN Bus Problems
Entry	VARAN not-required-frame error OR VARAN not-required-frame error DO:E80000BC
Description	An access error has occurred in a non-required module.
Cause / Solution	It can occur during the system shutdown. This can occur during shut down or disconnection of a non-required module. Hardware problems: See chapter Troubleshooting VARAN Bus Problems
Entry	VARAN fatal error DO:E8000724
Description	Operating error in the software of the VARAN Manager or a collision of data packets in the VARAN bus.
Cause / Solution	It can occur during the system shutdown. Hardware problems: See chapter Troubleshooting VARAN Bus Problems
Entry	VARAN time-slice Error Interrupt
Description	The real-time task in the hardware classes is not yet finished with processing the DPRAM data when the VARAN Manager wants to start its real-time task. This entry is only valid if after it is made, "runStatus changed from 0 to 44" or "runStatus changed from RUN RAM(0) to VARAN MANAGER ERROR (44)" and a dump (as with an Exception error) is entered.
Cause / Solution	It can occur during the system shutdown. This can occur while ending the application (Reset, Runtime, Division by 0,). Time problem in the application's real-time program (RtWork). Optimize the application's RtWork. Set all RtWork in application to a lower clock time.

Install higher capacity CPU.							
Entry	VARAN;1;timeout while waiting for admin task						
Description	The Manager's administration task has not been completed after a timeout has elapsed.						
Cause / Solution	Power On/Off and contact SIGMATEK This can occur in combination with a "VARAN fatal error".						
Entry	VARAN Error L(3),P:010200 OR VARAN Error L(3),P:01-02-00						
Description	An access error has occurred in a VARAN module. The module is connected in the 3rd level (L(3)) => 2 modules between the CPU and the defective module. The module is connected to the Manager at port 1, on the following device, port 2 and on the following device, port 0. Note: In the logfile, port 0 corresponds to VARAN 1 in the hardware, Port 1 corresponds to VARAN 2,...						
Cause / Solution	Hardware problems: See chapter Troubleshooting VARAN Bus Problems						
Entry	VARAN,EPN:SN01844148 OR VARAN,EAN:SN01996774						
Description	An access error has occurred in the VARAN module with the serial number 01844148. Always in connection with "VARAN required-frame error" or "VARAN not-required-frame error"						
	<table border="0"> <tr> <td>EPN</td><td>The node address is not set (because it is not connected.)</td></tr> <tr> <td>EAN</td><td>The node has a valid address (node was connected up to now).</td></tr> </table>	EPN	The node address is not set (because it is not connected.)	EAN	The node has a valid address (node was connected up to now).		
EPN	The node address is not set (because it is not connected.)						
EAN	The node has a valid address (node was connected up to now).						
Cause / Solution	Hardware problems: See chapter Troubleshooting VARAN Bus Problems						
Entry	VARAN,EAN:V1,D1004,P0,L1						
Description	Always in connection with "VARAN required-frame error" or "VARAN not-required-frame error". Details of the VARAN module in which the error has occurred.						
	<table border="0"> <tr> <td>D</td><td>DeviceID</td></tr> <tr> <td>L</td><td>Hierarchical level in the VARAN Network</td></tr> <tr> <td>P</td><td>Port number</td></tr> </table>	D	DeviceID	L	Hierarchical level in the VARAN Network	P	Port number
D	DeviceID						
L	Hierarchical level in the VARAN Network						
P	Port number						

	V VendorID
Cause / Solution	Hardware problems: See chapter Troubleshooting VARAN Bus Problems
Entry	VARAN;1;VM0; SPIMasterReadList;DOL Header;iRet=-26 VARAN;1;VM0; SPIMasterProcessIDs;List Header,PageAddr=0x0007FF84,ListID=4 VARAN;1;VM0; SPIMasterProcessIDs;List Header;iRet=-26 VARAN;1;VM0; SPIMasterGetIDs;Process IDs;iRet=-26 VARAN;1;VM0; ConnectDevice;Address=0x00050000 VARAN;1;VM0; ConnectDevice;Tree:02 VARAN;1;VM0; Error no connection, VID=1,DID=1072,SN=01933607,iRet=-26 VARAN;1;VM0; Tree Topology: 0
Description	An Error has occurred during the module configuration phase. Description: "iConnectDevice;Tree:02" The module is connected to the Manager at port 0, on the following device, port 2 and on the following device, port . Note: In the logfile, port 0 corresponds to VARAN 1 in the hardware, Port 1 corresponds to VARAN 2, ...
Cause / Solution	The module was connected for only a short time. Hardware problems: See chapter Troubleshooting VARAN Bus Problems

8.2.6 FTP

Entry	FTPSVR;3;Initializing FTP-Server interface 01.01.003
Description	The FTP server was started (server version 01.01.003)
Cause / Solution	INFORMATION
Entry	FTPSVR;1;START:Read data from C:\ftps.ini failed, rc=-1008
Description	The C:\ftps.ini initialization file is not available.
Cause / Solution	Copy the ftps.ini file into the directory C:\.

8.2.7 VNC

Entry	VNCUSR;1; VNCsvr: Created VNC server task!
Description	Remote control/ remote diagnosis was started.
Cause / Solution	Info

8.2.8 Log19

Entry	Loader V02.02.092
Description	Version number of the loader
Entry	Default value defines SRAM format 1
Description	This message shows that SRAM format 2 was selected as a result of the default value.
Entry	Platform 32 defines SRAM format 2
Description	This message shows that SRAM format 2 was selected due to the platform with the identification number 32. There are 2 different SRAM formats: <ul style="list-style-type: none">• In format 1, all null-voltage protected objects are in the null-voltage protected SRAM• In format 2, the non-editable data of null-voltage protected objects in the C:\RAMFILE.DAT file and the editable data of the null-voltage protected objects are found in the null-voltage protected SRAM
Entry	Checking Sram
Description	Informs the user that the consistency test of the data structure of the null-voltage protected data is currently running.
Entry	Reorganization of the null-voltage protected data area
Description	To avoid having objects from different projects in the null-voltage protected data area that are no longer deleted, this data area is reorganized. Thereby, all objects from this data area are first written in a file (copy). All objects are then deleted from the data area. While initializing the null-voltage protected data objects in the project, a new entry is created in the data structure of the null-voltage protected data area. The value of the null-voltage protected data objects is then taken from the copy, if it is located therein, otherwise the initial value from the project is used. Thereby, more objects are available in the current project after initializing in the null-voltage protected data area.
Entry	MakeSramKopie: header written to C:\LSLDATA\SRAM.CPY (20 bytes)

Description	Message indicating that 20 bytes of the header were written in the copy-file of the null-voltage protected data.
Entry	MakeSramKopie: no copy created, because sram is not valid; trying to load old copy
Description	No copy-file of the null-voltage protected data was written since it is invalid An attempt is made to read an existing copy-file of the null-voltage protected data.
Entry	LoadSramKopie: header read from C:\LSLDATA\SRAM.CPY (20 bytes)
Description	Message indicating that 20 bytes of the header were read from the copy-file of the null-voltage protected data.
Entry	MakeSramKopie: data written to C:\LSLDATA\SRAM.CPY (72 bytes)
Description	Message indicating that 72 bytes of the data section were written in the copy-file of the null-voltage protected data.
Entry	LoadSramKopie: data read from C:\LSLDATA\SRAM.CPY (8096 bytes)
Description	Message indicating that 8096 bytes of the data section were read from the copy-file of the null-voltage protected data.
Entry	MakeSramKopie: no need to write to C:\LSLDATA\RAMFILE.CPY (crc32=0x00000000 and udEntries=0 are still the same)
Description	Message indicating that the RAMFILE.DAT file did not have to be newly created, as the existing copy has the same checksum.
Entry	MakeSramKopie succeeded, marking sram as invalid.
Description	Message indicating that the copy-file of the null-voltage protected data has been successfully written and the contents of the null-voltage protected data area have been marked as invalid.
Entry	Info: re-using binary descr.block-info from prev.run
Description	Message indicating that the data structure of the descriptor blocks from classes and objects did not have to be newly created. Instead, they can be restored from a file that was created in the previous project.
Entry	SramSaveFile: no need to write to ramfile.dat (crc32=0x40AE06DF and udEntries=0 are still the same)
Description	Message indicating that the RAMFILE.DAT file did not have to be newly created, as the existing file has the same checksum.
Entry	Sram is now valid.
Description	The data structure of the null-voltage protected data area is now valid

Entry	Sram-cells - found in copy:0, found in sram:0, not found: 0
Description	Message indicating where the elements for the data structure of the null-voltage protected (SRAM cells) must be loaded during configuration.
Entry	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 = 000FFFC0 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
Description	Error detected during the data structure consistency test of the null-voltage protected data area. All null-voltage protected data objects are loaded with the contents of a previously created copy (if a copy is available) or they are loaded with the initial value defined in the project.
Entry	LoadSramKopie failed, rc=-3, errCode = 0x2C
Description	Loading the file used as a clipboard while reorganizing the SRAM did not function.

8.3 CPU Status and Error Messages

Status and error messages are shown in the status test of the Lasal Class software. If the CPU has a status display (7-segment display), the status or error number is also shown here as well.



In addition, the status and/or error number on the terminal screen is shown in clear text.



Number	Message	Definition	Cause/solution
00	RUN RAM	The user program is currently running in RAM. The display is not affected.	Info
01	RUN ROM	The user program stored in the program memory module has been loaded into the RAM is currently running. The display is not affected.	Info
02	RUNTIME	The total time for all cyclic objects exceed the maximum time; the time can be configured using two system variables: <ul style="list-style-type: none">• Runtime: time remaining• SWRuntime: pre-selected value for the runtime counter	Optimize the application's cyclic task. Use higher capacity CPU Configure preset value
03	POINTER	Incorrect program pointers were detected before running the user program	Possible causes <ul style="list-style-type: none">• The program memory module is missing, not programmed or defect.• The program in the user program memory (RAM) is not executable.• The buffering battery has failed.• The user program has overwritten a software error. Solution

			<ul style="list-style-type: none"> Reprogram the memory module, if the error reoccurs exchange the module. Exchange the buffering battery Correct programming error
04	CHKSUM	An invalid checksum was detected before running the user program.	Cause/solution: s. POINTER
05	Watchdog	The program was interrupted through the watchdog logic.	<p>Possible causes</p> <ul style="list-style-type: none"> Interruptions blocked by the user program for an extensive period of time (STI instruction forgotten). Programming error in a hardware interrupt. INB, OUTB, INW, OUTW instructions used incorrectly. The processor is defect. <p>Solution</p> <ul style="list-style-type: none"> Correct programming error. Exchange CPU.
06	GENERAL ERROR	<p>GENERAL ERROR</p> <p>An error has occurred while stopping the application over the online interface.</p>	The error occurs only during the development of the operating system.
07	PROM DEFECT	An error has occurred while programming the memory module.	<p>Possible causes</p> <ul style="list-style-type: none"> The program memory module is defect. The user program is too large. The program memory module is missing. <p>Solution</p> <ul style="list-style-type: none"> Exchange the program memory module
08	RESET	<p>The CPU has received the reset signal and is waiting for further instructions.</p> <p>The user program is not processed.</p>	Info
09	WD DEFEKT	The hardware monitoring circuit (watchdog logic) is defect.	<p>Solution</p> <ul style="list-style-type: none"> Exchange CPU.

		After power-up, the CPU checks the watchdog logic function. If an error occurs during this test, the CPU deliberately enters an infinite loop from which no further instructions are accepted.	
10	STOP	The program was stopped by the programming system.	
11	PROG BUSY	reserved	
12	PROGRAM LENGTH	reserved	
13	PROG END	The memory module was successfully completed.	Info
14	PROG MEMO	The CPU is currently programming the memory module.	Info
15	STOP BRKPT	The CPU was stopped by a breakpoint in the program.	Info
16	CPU STOP	The CPU was stopped by the programming software.	Info
17	INT ERROR	The CPU has triggered a false interrupt and stopped the user program or has encountered an unknown instruction while running the program.	<p>Possible causes</p> <ul style="list-style-type: none"> • A nonexistent operating system was used. • Stack error (uneven number of PUSH and POP instructions). • The user program was interrupted by a software error. <p>Solution</p> <ul style="list-style-type: none"> • Correct programming error.
18	SINGLE STEP	The CPU is in single step mode and is waiting for further instructions.	Info
19	READY	A module or project was sent to CPU and is ready to run the program	Info
20	LOAD	The program is stopped and the CPU is currently receiving a new module or project.	Info
21	UNZUL. MODULE	The CPU has received a module that does not belong to the project.	<p>Solution</p> <ul style="list-style-type: none"> • Recompile and download the entire project

22	MEMORY FULL	The operating system memory /Heap) is too small. No memory could be reserved while calling an internal or interface function is called from the application.	<p>Possible causes</p> <ul style="list-style-type: none"> Memory is only allocated but not released. <p>Solution</p> <ul style="list-style-type: none"> Free memory
23	NOT LINKED	When starting the CPU, a missing module or a module that does not belong the project was detected.	<p>Solution</p> <ul style="list-style-type: none"> Recompile and download the entire project
24	DIV BY 0	A division error has occurred.	<p>Possible causes</p> <ul style="list-style-type: none"> Division by 0. The result of a division does not fit in the result register. <p>Solution</p> <ul style="list-style-type: none"> Correct program error
25	DIAS ERROR	While accessing a DIAS module, an error has occurred.	Hardware problems: See chapter Troubleshooting DIAS Bus Problems
26	WAIT	The CPU is busy.	Info
27	OP PROG	The operating system is currently being reprogrammed.	Info
28	OP INSTALLED	The operating system has been reinstalled.	Info
29	OS TOO LONG	The operating system cannot be loaded; too little memory.	Restart; report error to Sigmatek.
30	NO OPERATING SYSTEM	Boot loader message. No operating system found in RAM.	Restart; report error to Sigmatek.
31	SEARCH FOR OS	The boot loader is searching for the operating system in RAM.	Restart; report error to Sigmatek.
32	NO DEVICE	reserved	
33	UNUSED CODE	reserved	
34	MEM ERROR	The operating system loaded does not match the hardware configuration.	Use the correct operating system version
35	MAX IO	reserved	
36	MODULE LOAD ERROR	The LASAL Module or project cannot be loaded.	<p>Solution</p> <ul style="list-style-type: none"> Recompile and download the entire project

37	BOOTIMAGE FAILURE	A general error has occurred while loading the operating system.	Contact SIGMATEK
38	APPMEM ERROR	An error has occurred in the application memory (user heap).	Solution <ul style="list-style-type: none"> Correct allocated memory access error
39	OFFLINE	This error does not occur in the control.	This error code is used in the programming system to show that there is no connection to the control.
40	APPL LOAD	reserved	
41	APPL SAVE	reserved	
42	ETHERCAT MANAGER ERR	An error number was entered In the EtherCAT manager and stopped the program.	
43	ETHERCAT ERROR	A required EtherCat client was disconnected or communication error has occurred.	
44	VARAN MANAGER ERROR	An error number was entered In the VARAN manager and stopped the program.	Solution <ul style="list-style-type: none"> Read logfile
45	VARAN ERROR	A required VARAN client was disconnected or communication error has occurred.	Solution <ul style="list-style-type: none"> Read logfile Error Tree
46	APPL-LOAD-ERROR	An error has occurred while loading the application.	Possible causes <ul style="list-style-type: none"> Application was deleted. Solution <ul style="list-style-type: none"> Reload the application into the control.
47	APPL-SAVE-ERROR	An error has occurred while attempting to save the application.	
50	ACCESS-EXCEPTION-ERROR	A read or write access of a restricted memory area. (I.e. writing to the NULL pointer).	Solution <ul style="list-style-type: none"> Correct application errors
51	BOUND EXCEEDED	An exception error has occurred when accessing arrays. The memory area was overwritten through accessing an invalid element.	Solution <ul style="list-style-type: none"> Correct application errors
52	PRIVILEGED INSTRUCTION	An unauthorized instruction for the current CPU level was given. For example, setting the segment register.	Possible causes

			<ul style="list-style-type: none"> The application has overwritten the application program code. <p>Solution</p> <ul style="list-style-type: none"> Correct application errors
53	FLOATING POINT ERROR	An error has occurred during a floating-point operation.	
60	DIAS-RISC-ERROR	Error from the Intelligent DIASMaster.	Restart; report error to Sigmatek.
64	INTERNAL ERROR	An internal error has occurred, all applications are stopped.	Restart; report error to Sigmatek.
65	FILE ERROR	An error has occurred during a file operation.	
66	DEBUG ASSERTION FAILED	INTERNAL ERROR	Restart; report error to Sigmatek.
67	REALTIME RUNTIME	<p>The total time for all real time objects exceeds the maximum time allowed. The time cannot be configured.</p> <p>2 ms for 386 CPUs</p> <p>1 ms for all other CPUs</p>	<p>Solution</p> <ul style="list-style-type: none"> Optimize the application's realtime task (RtWork). Reduce the clock time for the real-time task of all objects. Correct application errors CPU is overloaded in real-time => use a higher capacity CPU.
68	BACKGROUND RUNTIME	<p>The total time for all background objects exceed the maximum time; the time can be configured using two system variables:</p> <p>-BTRuntime: time remaining</p> <p>-SWBTRuntime: pre-selected value for the runtime counter</p>	<p>Solution</p> <ul style="list-style-type: none"> Optimize the application's background task (background) Use higher capacity CPU Set SWBTRuntime correctly.
72	S-DIAS ERROR	A connection error with an S-DIAS module has occurred.	<p>Possible causes</p> <ul style="list-style-type: none"> real network does not match the project S-DIAS client is defective <p>Solution</p> <ul style="list-style-type: none"> analyze logfile
95	USER DEFINED 0	User-definable code.	
96	USER DEFINED 1	User-definable code.	

97	USER DEFINED 2	User-definable code.	
98	USER DEFINED 3	User-definable code.	
99	USER DEFINED 4	User-definable code.	
100	C_INIT	Initialization start; the configuration is run.	
101	C_RUNRAM	The LASAL project was successfully started from RAM.	
102	C_RUNROM	The LASAL project was successfully started from ROM.	
103	C_RUNTIME		
104	C_READY	The CPU is ready for operation.	
105	C_OK	The CPU is ready for operation.	
106	C_UNKNOWN_CID	An unknown object from a stand-alone or embedded object, or an unknown base class was detected.	
107	C_UNKNOWN_CONSTR	The operating system class cannot be created; the operating system is probably wrong.	
108	C_UNKNOWN_OBJECT	Indicates an unknown object in an interpreter program; more the one DCC080 object.	
109	C_UNKNOWN_CHNL	The hardware module number is greater than 60.	
110	C_WRONG_CONNECTION	No connection to the required channels.	
111	C_WRONG_ATTR	Wrong server attribute.	
112	C_SYNTAX_ERROR	Non-specific error. Recompile and download all project sections.	
113	C_NO_FILE_OPEN	An attempt was made to open an unknown table.	
114	C_OUTOF_NEAR	Memory allocation error	
115	C_OUT_OF_FAR	Memory allocation error	
116	C_INCOMPATIBLE	An object with the same name already exists but has a different class.	
117	C_COMPATIBLE	An object with the same name and class already exists but must be updated.	

224	LINKING	The application is currently linking.	
225	LINKING ERROR	An error has occurred while linking. An error message is generated in the LASAL status window.	
226	LINKING DONE	Linking is complete.	
230	OP BURN	The operating system is currently being burned into the Flash memory.	
231	OP BURN FAIL	An error has occurred while burning the operating system.	
232	OP INSTALL	The operating system is currently being installed.	
240	USV-WAIT	The power supply was disconnected; the UPS is active. The system is shutdown.	
241	REBOOT	The operating system is restarted.	
242	LSL SAVE		
243	LSL LOAD		
252	CONTINUE		
253	PRERUN	The application is started.	
254	PRERESET	The application is ended.	
255	CONNECTION BREAK		

8.3.1 Troubleshooting DIAS Bus Problems

Was the exact value of the 24 V measured?



Are all components supplied with power (DC OK LED or measure)?



 When the application is running, do the RX and TX LED (if provided) light or blink in all components?

 When the application is running, does the Synchronize LED (if provided) light in all components?

 Does the Reset LED (if provided, i.e. CIC) light when the application is running?

 Retries DIASMaster, DIV512, CIV521, ...?

 Retries only when the frequency converter or another noise source is running?

 Are terminating resistors installed correctly?

 Are terminating resistors double installed?

 Are any station numbers assigned more than once?



With different modules, the LOW and HIGH hex switches are inverted =>
Be sure to check the labeling



With repeaters, there is one input and one output (CIC012, DIC121,...)!



Maximum cable length exceeded?
Maximum 20 m without the repeater for Lappkabel / UNITRONIC BUS FD
P LD 2 x 2 x 0.25 mm²



Correct cable type?



Which DIAS error causes the CPU to crash (CPU or 7-segment display)?
With 7-segment displays, press the Set button to show the station number



How are the earth connections/shielding/control cabinet structures?



Are all components connected to earth (module carrier, DIN rail, CPU,...)?



Are all components recognized by the software?



Does the detected hardware configuration match the actual configuration?



Voltage breaks in the power module of the DIAS bus?

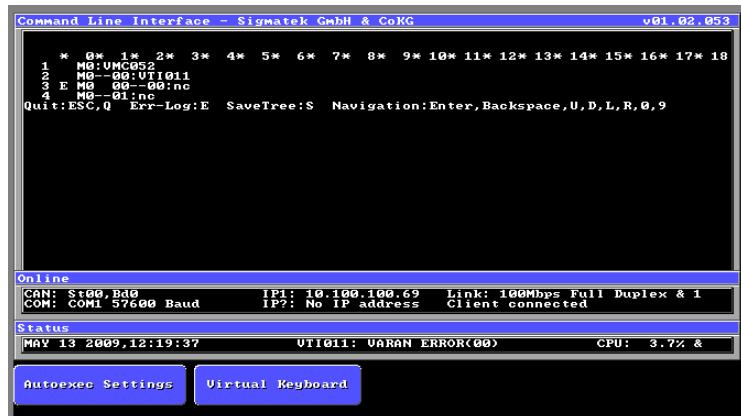


Module defective?

8.4 Troubleshooting for VARAN Bus Problems

8.4.1 Error Tree

With newer operating systems, the VARAN error tree is called automatically with CPUs that have a display. With the error tree, the module that caused the error can be identified. With some operating systems, the "T" button must first be pressed.



The numbers 0 to 18 and 1 to 4 serve only as navigation help for large VARAN networks. Navigation with buttons: Enter, Backspace, U, D, L, R, 0, 9

1st line: M0:VMC052:

Manager 0 has the description VMC052 (could also be: ETV1961-K, HGT833, ETV0811,...)

2nd line: M0-00:VTI011 :

On port 0 of manager 0, a VTI011 is connected.

3rd line: E M0 00-00 : nc

At port 0 of the VTI011, an error has occurred (E => Error)

In this case, the client on the downlink port of the VTI011 was disconnected while the application was running.



Using the error tree, it is possible to identify the accessed module that triggered the error, but this does not necessarily mean that exchanging the module will solve the problem. It only means the error can be limited to the path between the possibly defective module and the CPU. For this path, the hardware checklist must be used.

8.4.2 Hardware Checklist



Was the exact value of the 24 V measured?



Are all components supplied with power (DC OK LED or measure)?



Do Link (green LED) and Active (orange LED) blink in all components?



Does the Synchronize LED (if provided) light in all components?

 Does the Reset LED (if provided, i.e. C1V) light when the application is running?

 Read logfile

 Retries VARAN Manager
=> Check earth connections/shielding/control cabinet structures.

 Is the cable assembled correctly?
The cable length must be less than 100 m

 Module-specific retry counter
With the module-specific retry counter, poorly placed or incorrect twisted cables can be found. Retries only when the frequency converter or another noise source is running?
=> Check earth connections/shielding/control cabinet structures!

 With which VARAN error does the CPU stop (read logfile, CPU status on the screen of 7-segment display)?
=> Check earth connections/shielding/control cabinet structures!

 Are all components connected to earth (module carrier, DIN rail, CPU,...)?

 Have all components been recognized?



Does the detected hardware configuration match the actual configuration?



Voltage breaks on the module?



Module defective?

9 SIGMATEK Device Configuration

It is possible to configure SIGMATEK controllers using a web service. This chapter describes the corresponding procedure.



Attention!

The SIGMATEK Device Configuration is **only available under the operating systems Salamander / Gecko!**

- [General information](#)
- [The LOGIN window](#)
- [Areas in the Device Configuration](#)
- [Logout](#)
- [Troubleshooting](#)
- [Disclaimer](#)

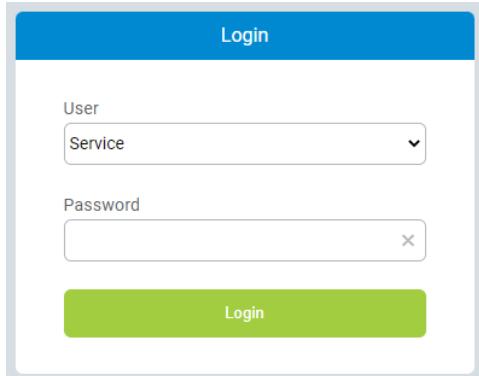
9.1 General Information

A SIGMATEK CPU with Salamander / Gecko operating system can be configured via web service. To do this, enter the IP address of the controller to be configured plus the preconfigured port in a browser. (By default it is preconfigured <http://10.10.150.1:1980>)

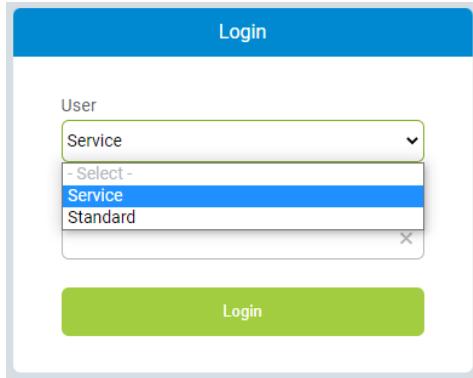
The created user configuration is stored on the controller as a file `C:\ls1sys\webconfigusers`. Such a configuration **can be transferred to other controllers by means of a USB stick, or via the Machine Manger Update Tool.**

9.2 The LOGIN window

If the SIGMATEK Device Configuration is called up for the first time, or if a "Reload" has been executed or the user has logged out, the LOGIN window is displayed:



In the LOGIN window in the SIGMATEK Device Configuration, you can choose between two predefined users: "[Service](#)" and "[Standard](#)".



Tip

The default password in the delivery state is "sigmatek" for both users.

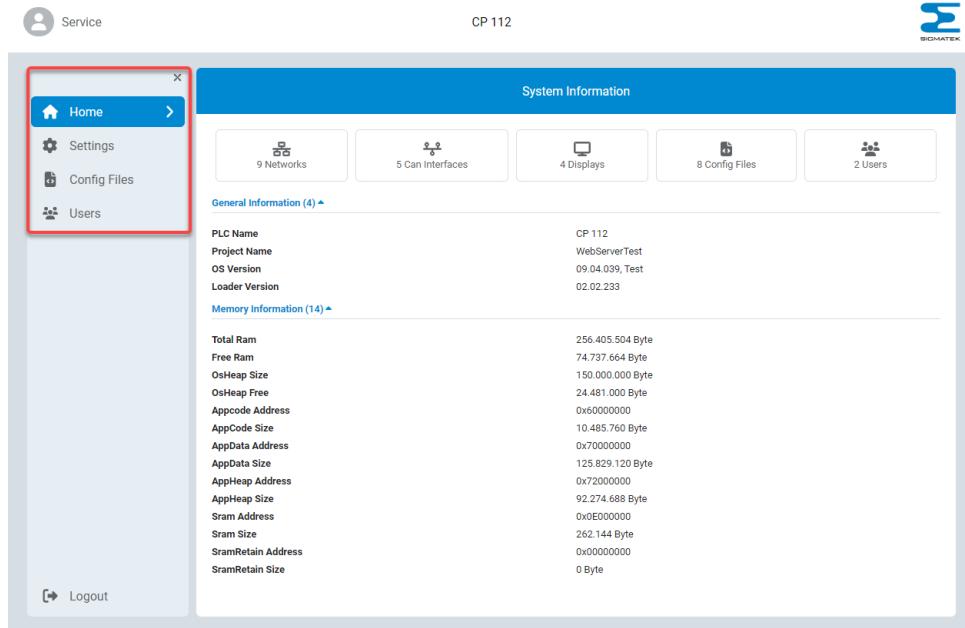
About the capabilities of the users, see the chapter [User Management](#)

For information on logging out of the SIGMATEK Device Configuration or applying the changes, see the [Logout](#) chapter.

9.3 Areas in the Device Configuration

If you are successfully logged in, you can start the configuration. There are a maximum of four areas on the left side, which can be displayed depending on the user authorization. If you are logged in with the user name "Service", all areas are displayed; if you are logged in with "Standard", only the first two areas are visible.

The areas can be displayed in three different ways, depending on the screen size and resolution. For high resolutions, on the left as icons with names:



The screenshot shows the Device Configuration interface for a CP 112 module. On the left, a sidebar for the "Service" user is visible, containing icons for Home, Settings, Config Files, and Users. The main area is titled "System Information" and displays the following data:

Category	Value
9 Networks	5 Can Interfaces
4 Displays	8 Config Files
2 Users	

Below this, the "General Information (4)" section shows:

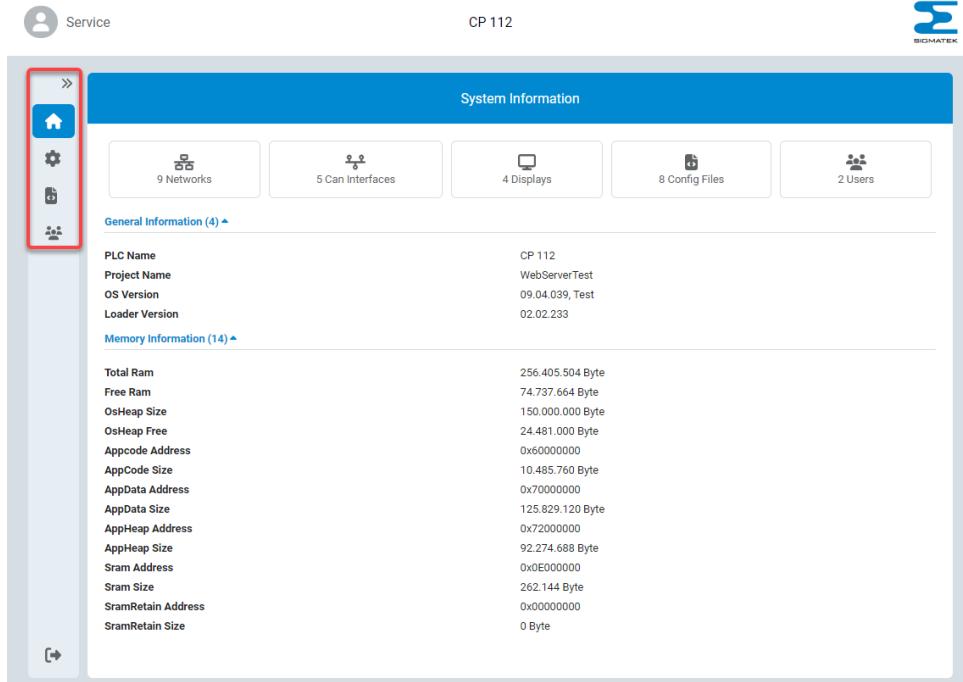
Parameter	Value
PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

The "Memory Information (14)" section shows:

Parameter	Value
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

At the bottom left of the main panel is a "Logout" button.

If you click on the "x" above the home button, the area display is "collapsed", i.e.: only the icons are visible, the descriptions disappear:



Service

CP 112

System Information

9 Networks 5 Can Interfaces 4 Displays 8 Config Files 2 Users

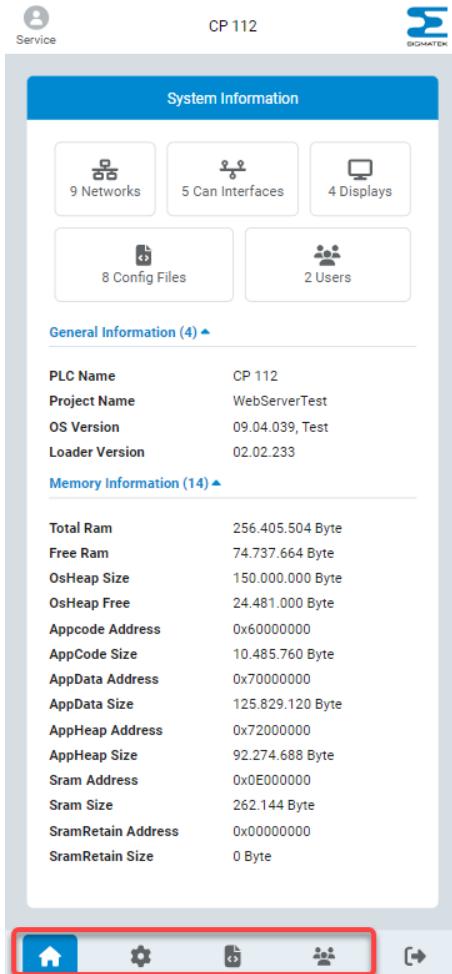
General Information (4) ▲

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

The double arrow above the home icon can be used to expand the area display again. If the display - e.g. for a mobile device - is in smaller resolution and portrait format, the icons "move" to the bottom:



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) ▲

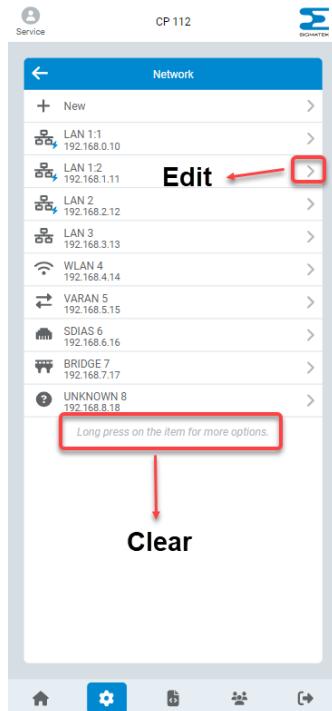
Total Ram	256.405.504 Byte
Free Ram	74.737.664 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
Appcode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	125.829.120 Byte
AppHeap Address	0x72000000
AppHeap Size	92.274.688 Byte
Sram Address	0x0E000000
Sram Size	262.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

Home    

Special operation in portrait mode

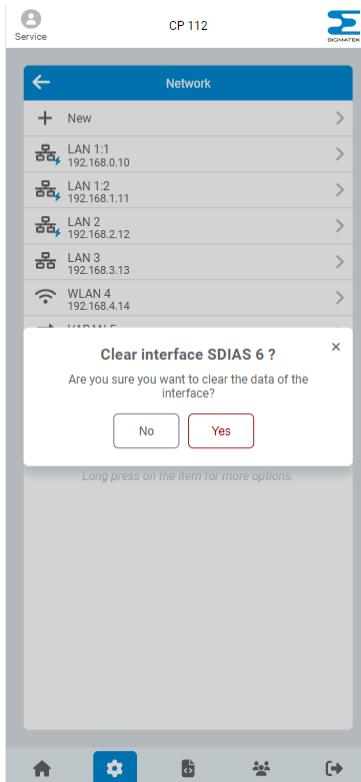
At this point, we would like to point out a **special feature of the operation** in this "mobile" portrait resolution. If an edit window (e.g. in the Settings under Network, Can Bus, or Display) is opened in this format, no "Clear" and "Edit" buttons can be displayed due to

the small width. Instead, a **More arrow** is displayed on the right side of each displayed line, as well as the note "Long press on the item for more options" at the bottom.



A short tap on the entry corresponds to the "Edit" button in the horizontal resolution.

A longer press corresponds to the "Clear" button in the horizontal resolution. The query dialog of the "Clear" button is also displayed immediately:

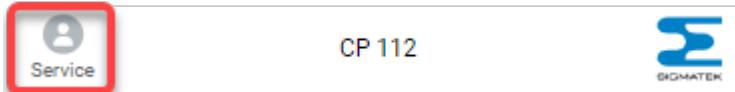


Now follows the description of the individual areas:

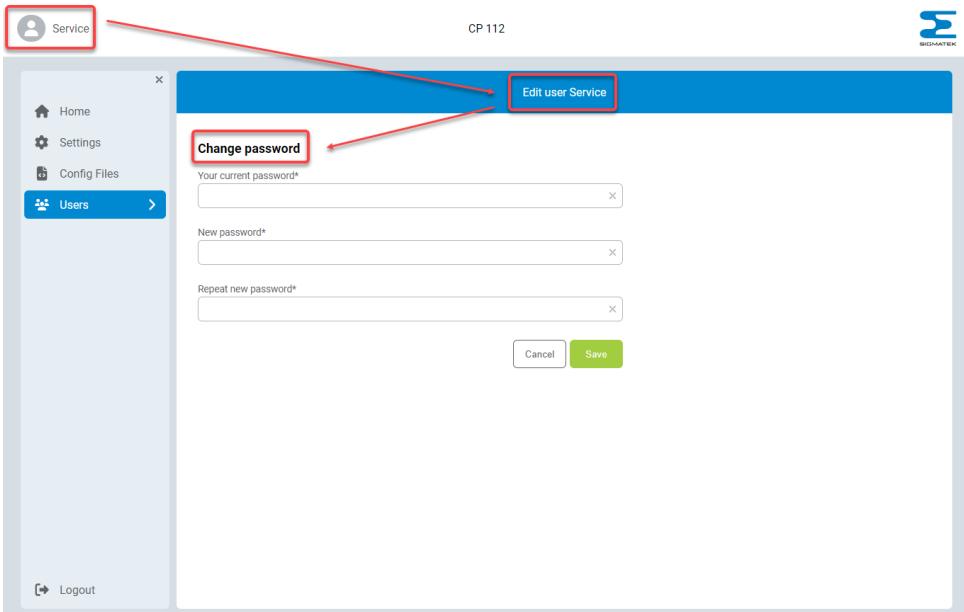
- [Title bar](#)
- [Home](#)
- [Settings](#)
- [Config Files](#)
- [Users](#)

9.3.1 Title bar

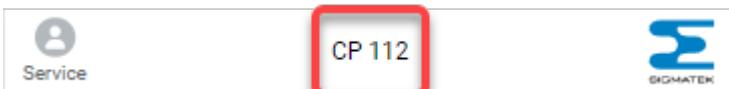
There are three icons in the title bar of the website.



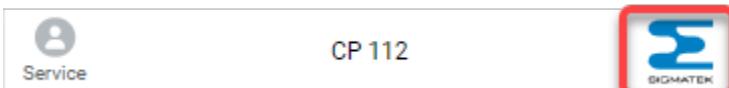
On the left you can see the currently logged in user. With a mouse click on the logged in user the dialog for the (own) password change is displayed:



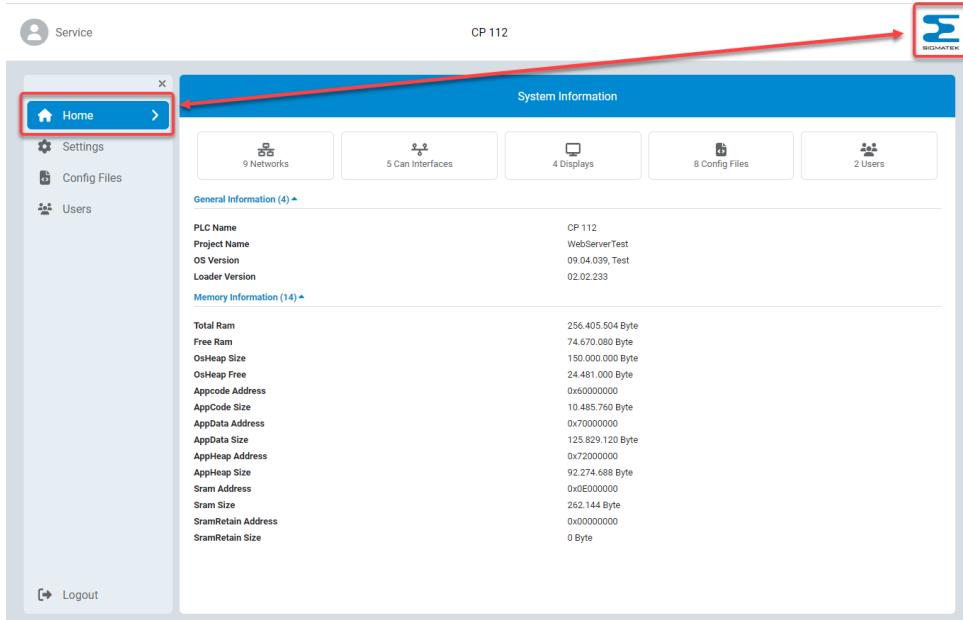
In the middle you can see the type (name) of the controller which should be configured:



The button on the right...



...corresponds to the button for the home area:



The screenshot shows the LASAL OS web interface. On the left, a sidebar menu includes 'Service', 'Home' (which is highlighted with a red box and has a red arrow pointing to it), 'Settings', 'Config Files', and 'Users'. The main content area is titled 'System Information' and shows 'CP 112' at the top. It includes sections for 'General Information' (4 items) and 'Memory Information' (14 items). The 'General Information' table is as follows:

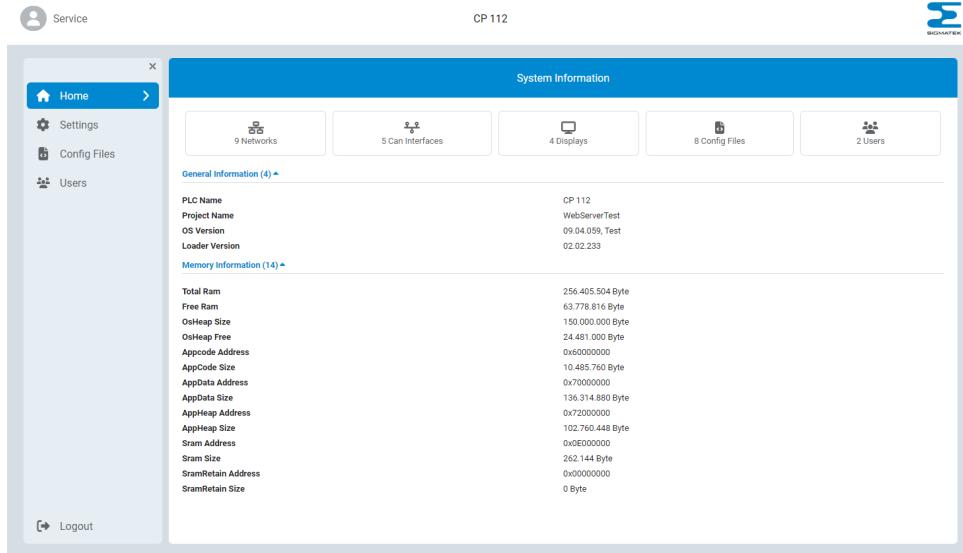
PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

The 'Memory Information' table is as follows:

Memory Type	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 screen. At the top, there is a user icon labeled "Service" and the identifier "CP 112". On the right, the SIGMATEK logo is visible. The main area is titled "System Information" and contains two sections: "General Information (4)" and "Memory Information (14)". The "General Information" section shows the following data:

PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.009, Test
Loader Version	02.02.233

The "Memory Information" section shows the following data:

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, there is a "Logout" button. On the left side, a sidebar lists "Home", "Settings", "Config Files", and "Users".

The home screen **shows information of the controller**, which can also be displayed in PlcDiag (Target Info) or LASAL CLASS 2 (Project Info) with an existing online connection. In addition, there are five fixed, preconfigured shortcuts in the upper area, which point to certain subitems in individual areas:



Attention!

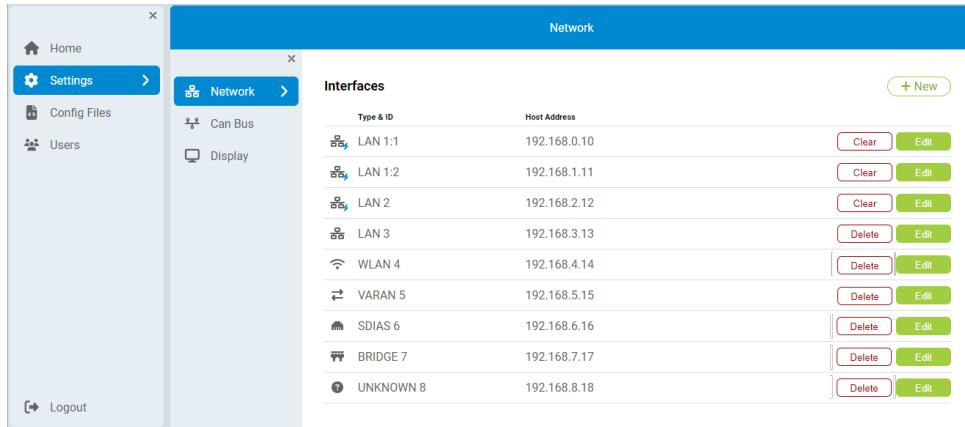


The "Standard" user only sees the first three shortcuts! For more information about the selectable users see the chapter [User Management](#)

The first three shortcuts call up the corresponding items in the Settings area [Settings \(Network, Can Bus, Display\)](#).

The last two shortcuts call the two areas ([Config Files](#), [Users](#)) that only the "Service" user sees.

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"/>

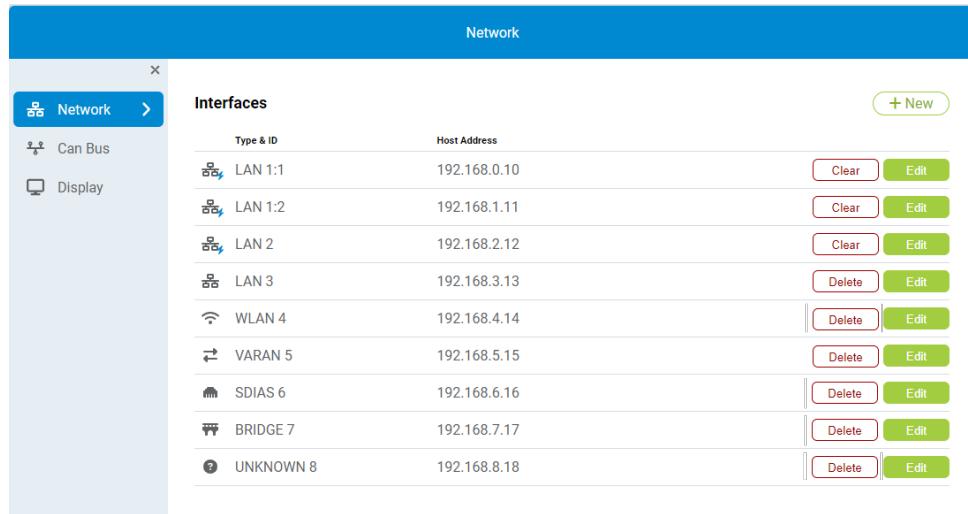
In the "Settings" area, three additional buttons can be clicked and used to configure [Network](#), [Can-Bus](#)- and [Display interfaces](#) konfiguriert werden. From the "Home" area, the first three fixed configured shortcuts also point to these pages.

Important!

All changes are only applied **after the system has been restarted**, which can be done either manually or in the [logout dialog](#) with the "**Reboot**" button.



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

Network interfaces can be configured and managed on this page. With the "x" above "Network", this second menu level can also be collapsed and expanded with the double arrow that then appears, as is possible in the top level. In the closed state, only the icons and no descriptions are displayed here.

A new network interface can be created using the "**+New**" button:

New network

Network > **+ Interface**

Can Bus

Display

Interface ID*

Type

use DHCP

Host Address*

Subnet*

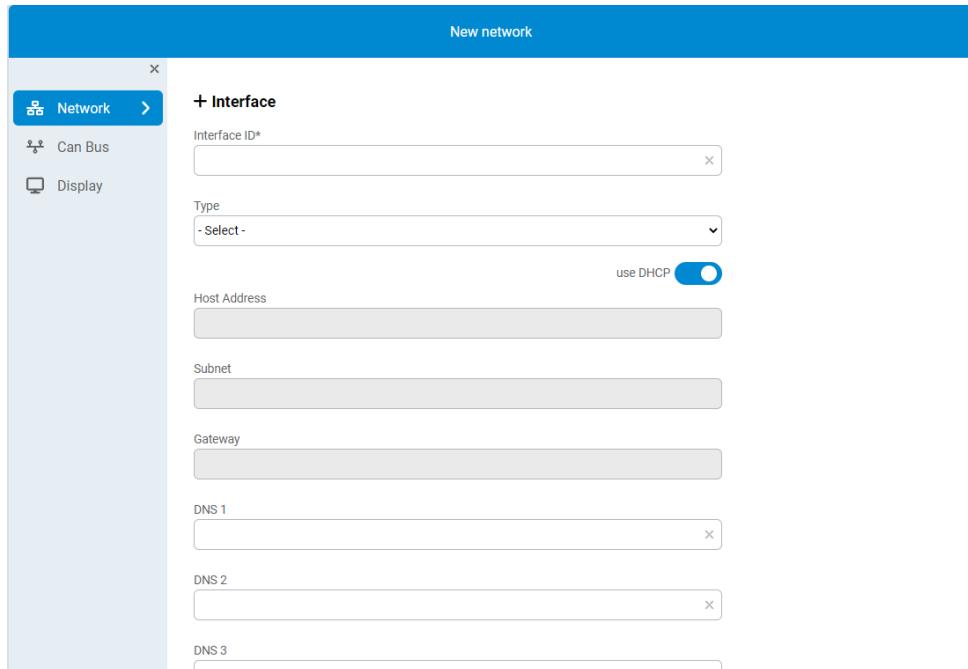
Gateway

DNS 1

DNS 2

DNS 3

As usual, fields marked with an asterisk (*) are mandatory. However, selecting "**use DHCP**" here can change the nature of the fields:



As you can see, fields that were mandatory in the other position of the "use DHCP" switch are disabled (grayed out), and are therefore no longer mandatory fields.

The ID for interfaces of the "LAN" type can either be one digit or two digits separated by a colon. A LAN interface can be assigned a maximum of two IP addresses, so it can be registered in two networks at the same time. In the first picture in this chapter LAN 1 is such an interface. The first number is the ID, the second the number of the network / IP address.

In the first picture you also notice that some icons in the column "Type & ID" have a small



blue flash in the lower right corner (), others do not. This blue flash is the **"attached" symbol**, so it means that the interface is actually **present as hardware** (in the CPU, or e.g. plugged in as a USB stick). The interfaces **without this symbol** have been configured, but are **not physically present**.

To the right of the "Type & ID" column, the host address and two further buttons ("Clear" or "Delete" and "Edit") are displayed. **"Clear" deletes the configuration of a physically existing interface** (with the lightning symbol) after a query - this can be recognized by the

fact that the corresponding **icon is displayed in light gray**, the column "Host Address" shows "**not configured**" instead of an IP address - and the "**Clear**" button **disappears**:

Network			
Interfaces			
	Type & ID	Host Address	
Can Bus	LAN 1:1	not configured	<input type="button" value="Edit"/>
Display	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"/>

For **physically non-existent interfaces** (without flash), there is no "**Clear**" button, but a "**Delete**" button. Clicking this **removes the interface completely**.

With the "**Edit**" button, the respective network interface can be configured. Exactly the same window is displayed as when clicking the "**+New**" button, except that the fields are **already filled with the current data**.

The first two fields (Interface ID and Type) **cannot be changed** for "**attached**" interfaces. Neither can the last field ("**Hardware Address**") be changed in such a case. It was already mentioned above that the three fields below the "**use DHCP**" button are disabled as soon as this button is activated.

With "**Cancel**" the configuration process **can be aborted**, with "**Save**" the **changes are accepted**.

Attention!

If "**Save**" is clicked, the selected settings **are written back to the Autoexec.lsl file**, so it should be configured with care, because it is possible that after a change no connection to the controller is possible anymore!



**Important!**

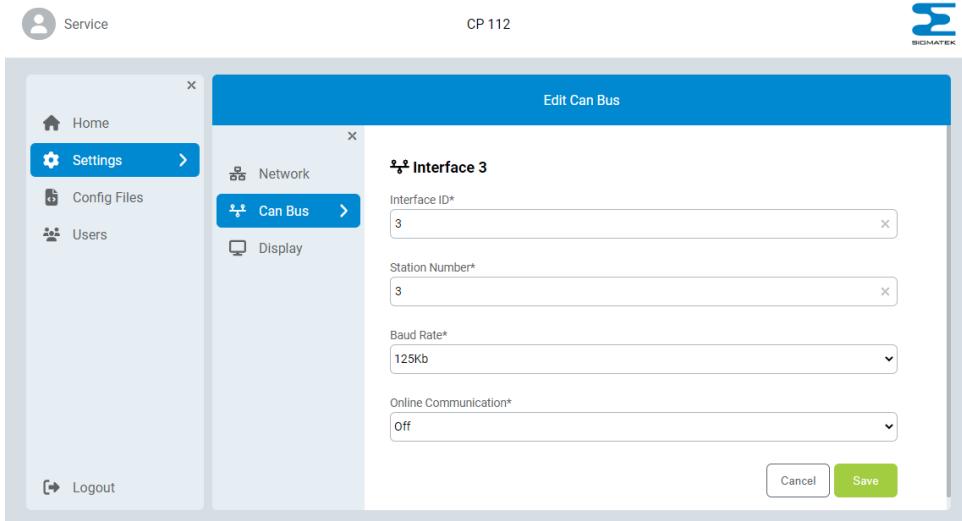
All changes are only applied **after the system has been restarted**, which can be done either manually or in the [logout dialog](#) with the "**Reboot**" button.

9.3.3.2 Can Bus

The management window for the configuration of Can Bus interfaces is very similar to the one for network interfaces.

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

The display ("blue flash" for actually existing, no flash for preconfigured, but physically not existing hardware) as well as the functionality of the buttons "**+New**", "**Clear**" or "**Delete**" and "**Edit**" are identical to those in the management of the network interfaces. Clearly the 2nd column (station number and baud rate) differs as well as the fields that can be entered when clicking on "**Edit**":



Service CP 112

Edit Can Bus

Interface 3

Interface ID*
3

Station Number*
3

Baud Rate*
125kb

Online Communication*
off

Cancel Save

ID, station number and baud rate are self-explanatory. The last field "**Online Communication**" determines whether you **should be able to go online from LASAL through this Can Bus interface** (On), or not (Off). Also on this page, the fields marked with an asterisk (*) are mandatory fields.

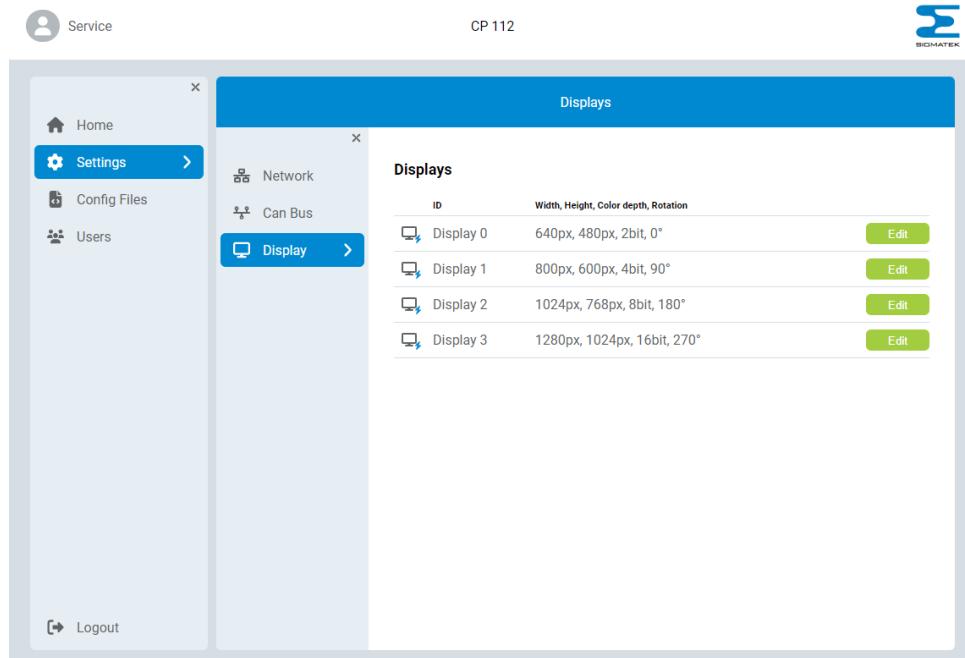


Important!

All changes are only applied **after the system has been restarted**, which can be done either manually or in the [logout dialog](#) with the "**Reboot**" button.

9.3.3.3 Display

The management window for the display interfaces (graphics cards) is also very similar in structure and operation to the previous two windows:



Service 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

Home

Settings >

Config Files

Users

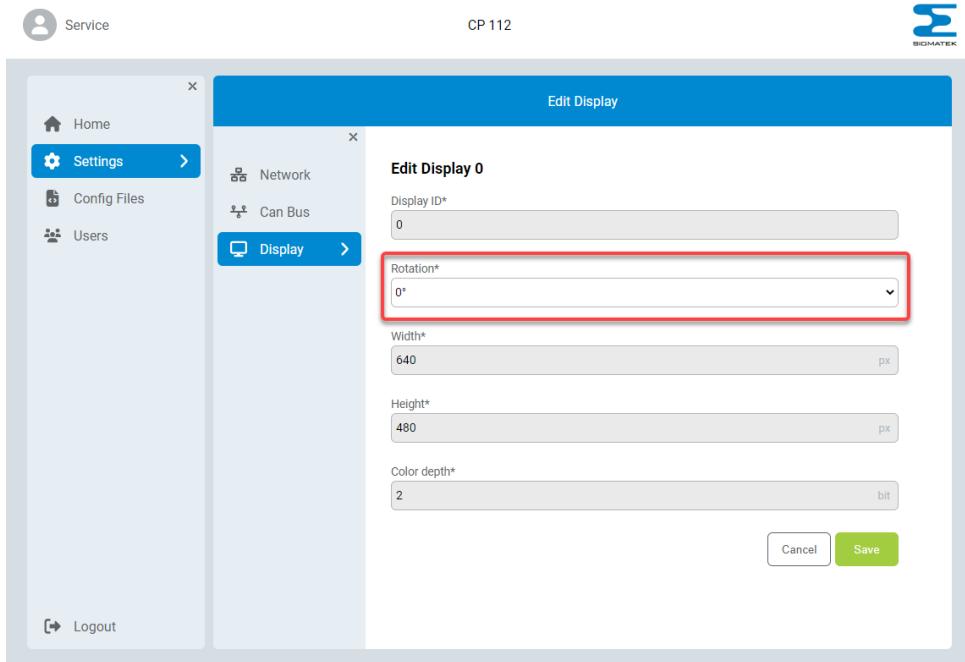
Network

Can Bus

Display >

Logout

Here it is noticeable that all interfaces have the "blue flash" - this is also one of the main differences to the other administration windows: only physically present displays (=graphics cards) are shown. Since these are recognized by the SIGMATEK Device Configuration itself, only one field can be changed after clicking on the Edit button, which is also a mandatory field: the rotation.



Service CP 112

Edit Display

Edit Display 0

Display ID*
0

Rotation*
0°

Width*
640 px

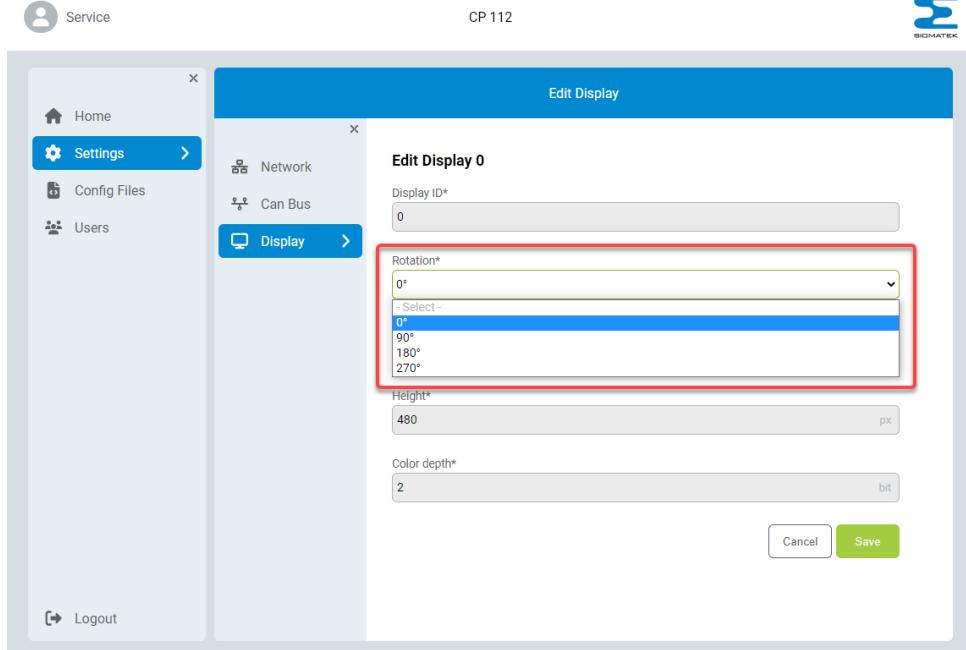
Height*
480 px

Color depth*
2 bit

Cancel Save

Logout

Here you can choose between the four rotations, each around a right angle:



Service CP 112

SIGMATEK

Edit Display

Edit Display 0

Display ID*

0

Rotation*

0°
-Select-
0°
90°
180°
270°

Height*

480 px

Color depth*

2 bit

Cancel Save

All other fields (ID, resolution, bit depth) are automatically recognized and entered.

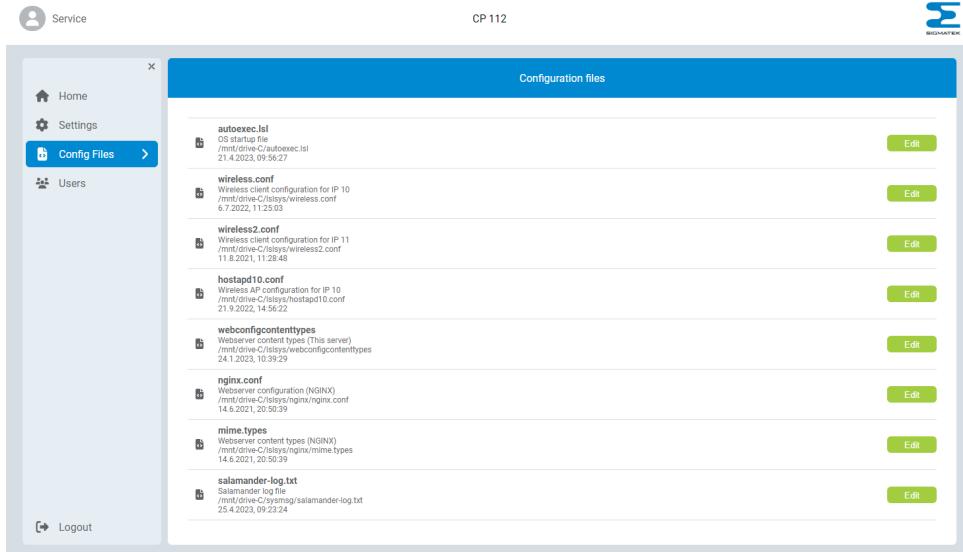
Important!

All changes are only applied **after the system has been restarted**, which can be done either manually or in the [logout dialog](#) with the "Reboot" button.



9.3.4 Config Files

This area is reserved for the "[Service](#)" user. Here the configuration files can be edited. Not all of these files must necessarily be present, some are optional.



CP 1112

Configuration files

File	Path	Modified	Size	Action
autoexec.lsl	/OS startup file /mnt/drive-C/lsys/autoexec.lsl	21.4.2023, 09:56:27	2.12 KB	Edit
wireless.conf	/Wireless client configuration for IP 10 /mnt/drive-C/lsys/wireless.conf	6.7.2022, 11:23:03	1.1 KB	Edit
wireless2.conf	/Wireless client configuration for IP 11 /mnt/drive-C/lsys/wireless2.conf	11.8.2021, 11:28:48	1.1 KB	Edit
hostapd.conf	/Wireless AP configuration for IP 10 /mnt/drive-C/lsys/hostapd10.conf	21.9.2022, 14:56:22	1.1 KB	Edit
webconfigcontenttypes	/Webserver content types (This server) /mnt/drive-C/lsys/webconfigcontenttypes	24.1.2023, 10:39:29	1.1 KB	Edit
nginx.conf	/Webserver configuration (NGINX) /mnt/drive-C/lsys/nginx/nginx.conf	14.6.2021, 20:50:39	1.1 KB	Edit
mime.types	/Webserver content types (NGINX) /mnt/drive-C/lsys/nginx/mime.types	14.6.2021, 20:50:39	1.1 KB	Edit
salamander-log.txt	/Salamander log file /mnt/drive-C/lsysmsg/salamander-log.txt	25.4.2023, 09:23:24	1.1 KB	Edit

If you click on the "Edit" button, the corresponding configuration file (here as an example the file Autoexec.lsl) can be changed and saved with the "Save" button:

Service

CP 313

SIGMATEK

Home

Settings

Config Files

Users

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 OSHEAP 120000

REM ### OPEN WEBCONFIG SECTION SET CAN
SET CAN_E_BAS 1 STARTED
REM ### CLOSE WEBCONFIG SECTION SET CAN

REM ### OPEN WEBCONFIG SECTION SET IP
SET IP 1 HOSTASDRA.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 ****

CALIB CHECK
SET RUNTIME 30
VER
LSLLLOAD
RUN
```

Cancel Save

Logout

Attention!



Extreme caution is required: all changes to the configuration files are **written directly to the controller!** At the next boot, these values are then valid, and it may be that e.g. no connection can be established to the controller any more!

If a new setting is entered in the Settings section on the "Network", or "Can Bus" page, it is inserted **at the end of the Autoexec.lsl file**. If you want to configure the position where the Network or Can Bus settings are inserted within the Autoexec.lsl file, the area for these settings can be enclosed (marked) by special comments as a "Section Block". These are for the "Network" page:

```
REM ### OPEN WEBCONFIG SECTION SET IP
```

```
...
```

```
REM ### CLOSE WEBCONFIG SECTION SET IP
```



and for the "Can Bus" page:

```
REM ### OPEN WEBCONFIG SECTION SET CAN
```

```
...
```

```
REM ### CLOSE WEBCONFIG SECTION SET CAN
```

If no valid "Section Block" is found for the setting, a message on the respective page in the Settings area points out this fact as well as the necessary syntax (in our example it is the "Network" page):

No valid section block was found in the 'autoexec.lsl'.

Please insert the following lines:

```
REM ### OPEN WEBCONFIG SECTION SET IP
```

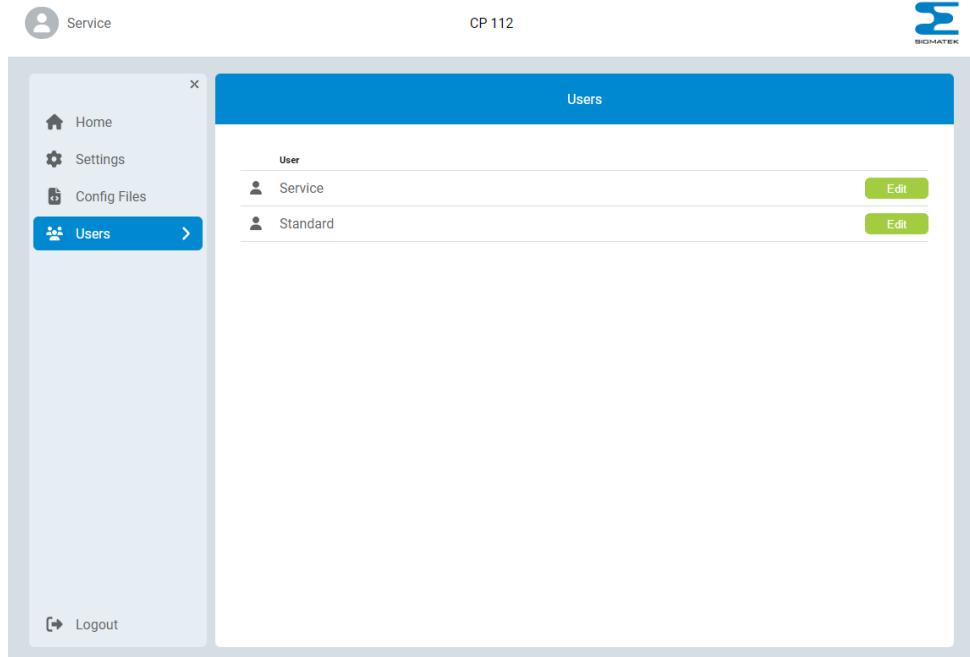
```
REM ...
```

```
REM ### CLOSE WEBCONFIG SECTION SET IP
```

Note: Without these lines new or changed interfaces will be inserted at the bottom of the configuration.

9.3.5 Users

This area is also reserved for the "[Service](#)" user. Here the users can be managed:

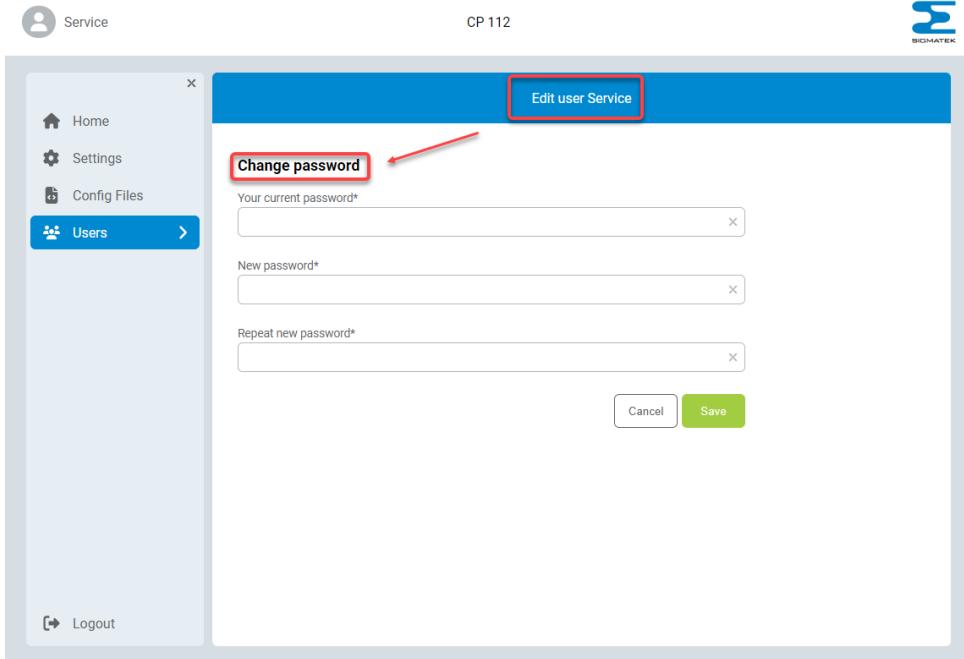


The screenshot shows a user interface for managing users. On the left, a sidebar menu includes 'Home', 'Settings', 'Config Files', and 'Users' (which is currently selected and highlighted in blue). On the right, the main content area is titled 'Users'. It displays a table with two rows:

User	
Service	Edit
Standard	Edit

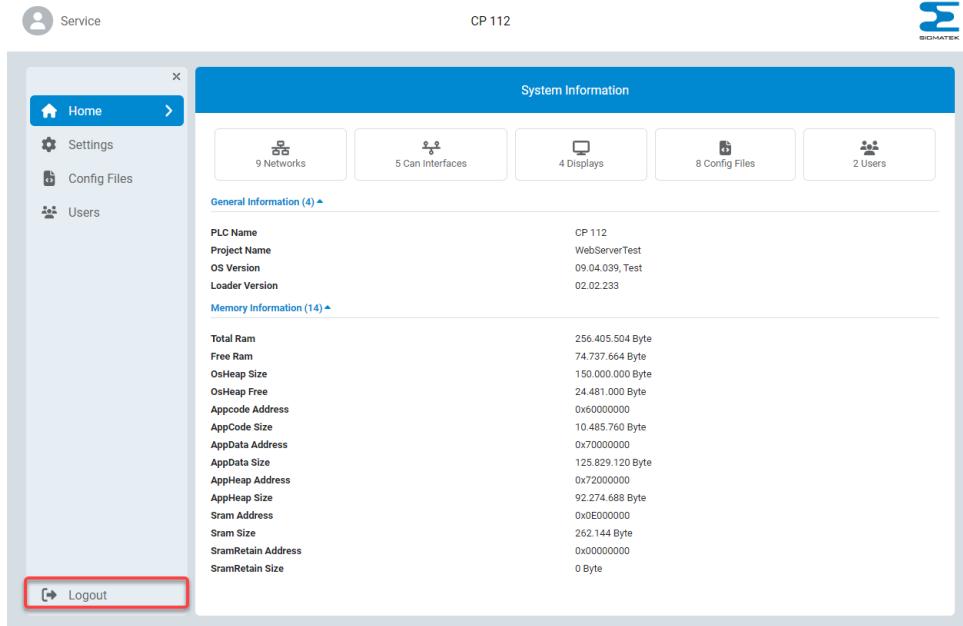
At the bottom left of the main area, there is a 'Logout' button.

"Manage" in this case means only to change the password:



9.4 Logout

If the configuration is completed, or if **another user is to be logged in**, the **web page reloaded** or the **controller rebooted**, the currently logged in user is **logged out**. This is done with the **logout button**, which is located on the left side, where the areas are displayed, at the very bottom (for left standing area displays)....



Service

CP 112

System Information

General Information (4) ▲

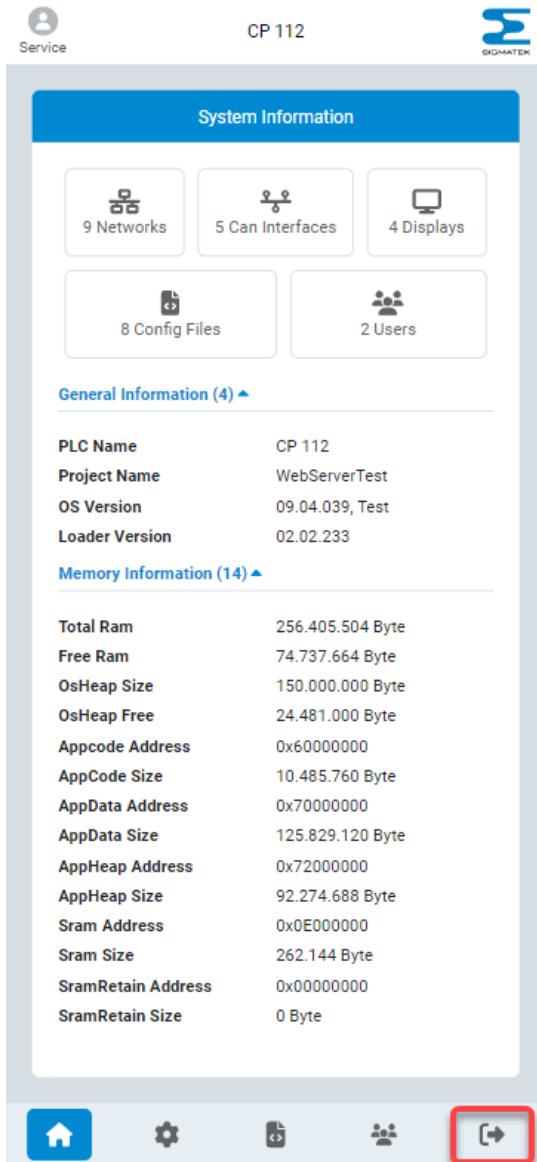
PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

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

...or on the far right (with area icons displayed below).



Service CP 112

System Information

- 9 Networks
- 5 Can Interfaces
- 4 Displays
- 8 Config Files
- 2 Users

General Information (4) ▲

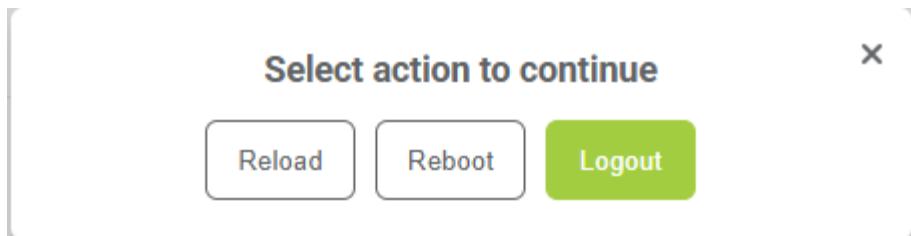
PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

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

Logout

If the **logout icon** is clicked, the following dialog appears:



Here you have three options.

- **Reload** reloads the web page
- With **Reboot** the control is **restarted**
- With **Logout** the **LOGIN dialog** appears and the user can log in again

Important!



All changes are only applied **after the system has been restarted**, which can be done either manually or here with the "**Reboot**" button.

9.5 User Management

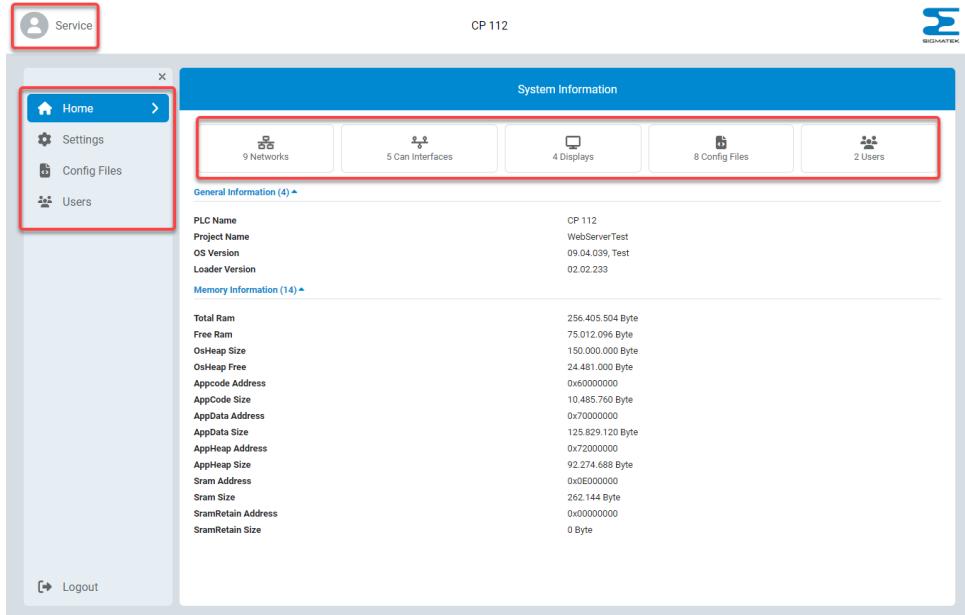
There are two predefined users in the SIGMATEK Device Configuration: "Standard" and "Service". What distinguishes the two and what they can do, you can read in the chapters

- [The "Service" user](#)
- [The "Standard" user](#)

9.5.1 The "Service" user

The user "Service" corresponds to an administrator. This user has all rights, i.e.:

- He can see all four areas ([Home](#), [Settings](#), [Config Files](#), [Users](#)) as well as all five possible shortcuts:



CP 112

System Information

General Information (4) ▾

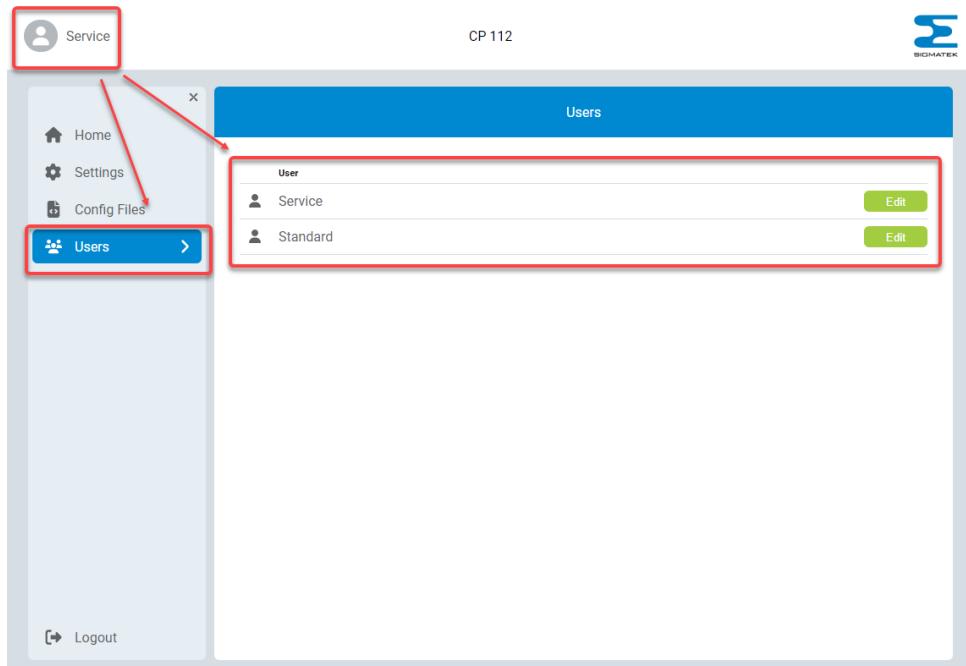
PLC Name	CP 112
Project Name	WebServerTest
OS Version	09.04.039, Test
Loader Version	02.02.233

Memory Information (14) ▾

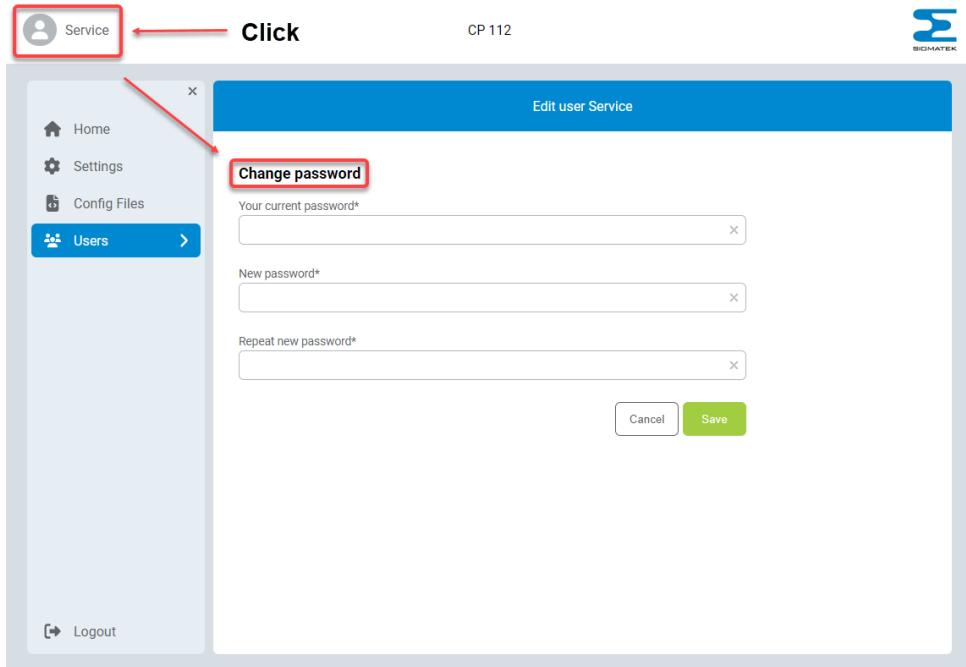
Total Ram	256.405.504 Byte
Free Ram	75.012.096 Byte
OsHeap Size	150.000.000 Byte
OsHeap Free	24.481.000 Byte
AppCode Address	0x60000000
AppCode Size	10.485.760 Byte
AppData Address	0x70000000
AppData Size	125.829.120 Byte
AppHeap Address	0x72000000
AppHeap Size	92.274.688 Byte
Sram Address	0x0E000000
Sram Size	252.144 Byte
SramRetain Address	0x00000000
SramRetain Size	0 Byte

Logout

- He can manage not only his own but also the "[Standard](#)" account, that is: change the password by clicking on the "[Users](#)" section:



Clicking on "Edit" for the "Service" user corresponds to clicking on the user name in the upper left corner:



Click

Service

Change password

Edit user Service

CP 112

Home

Settings

Config Files

Users

Logout

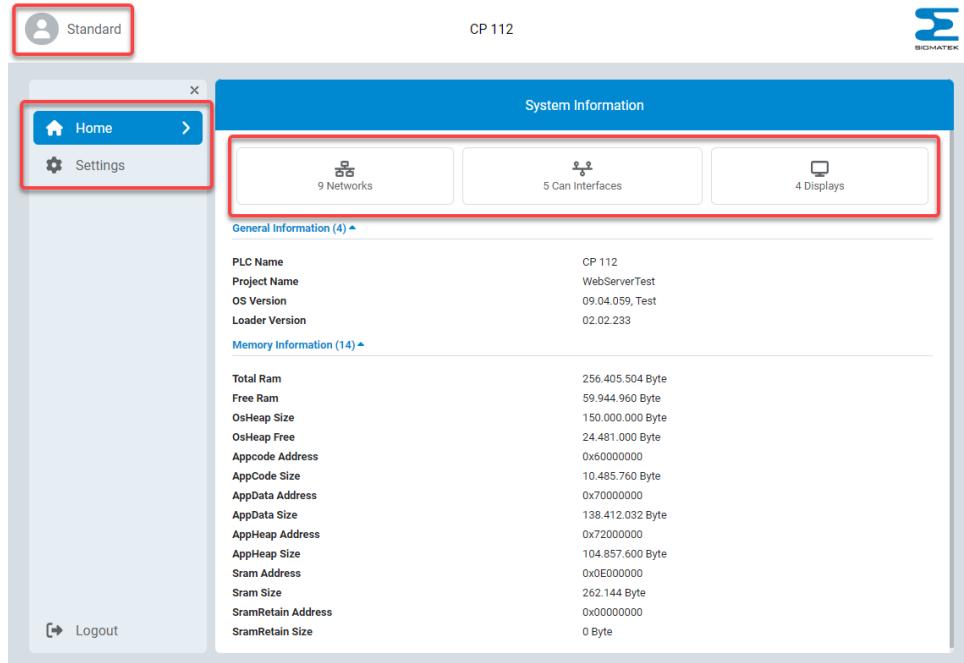
**Important!**

The password for the user "Service" is "sigmatek" in the factory setting

9.5.2 The "Standard" user

The user "Standard" has only limited rights, i.e.:

- He can only see the first two areas ([Home](#), [Settings](#)) and only three out of the five possible shortcuts:

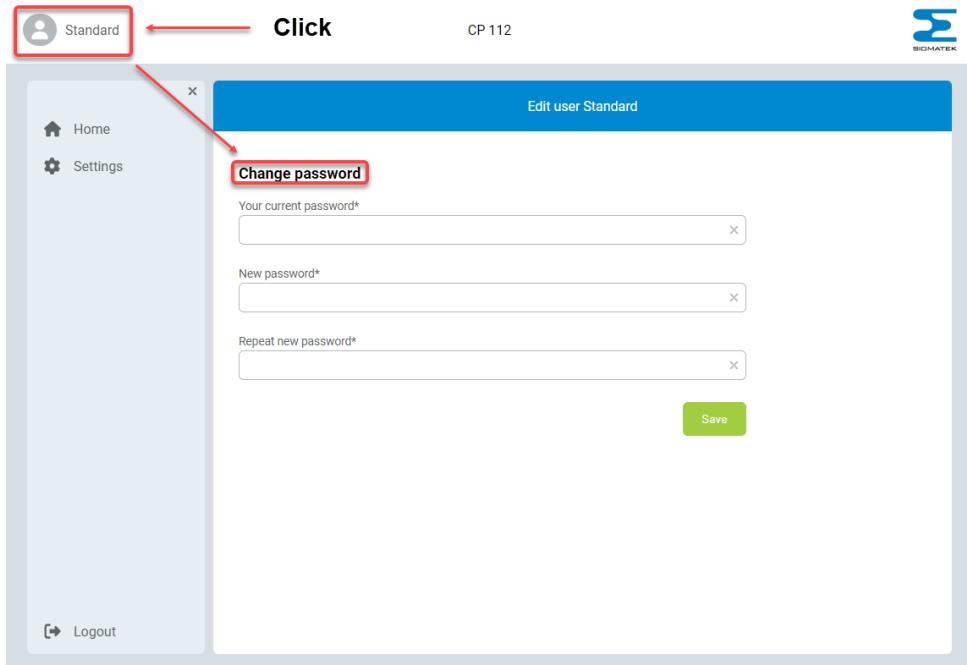


CP 112

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

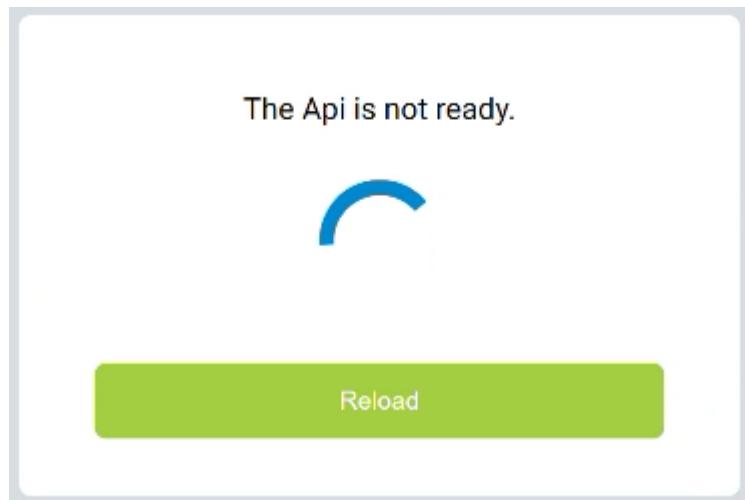
- He can only manage his own account, that is: change his own password:

**Important!**

The password for the user "Standard" is "sigmatek" in the factory setting

9.6 Troubleshooting

In order to configure a controller with SIGMATEK Device Configuration, it is necessary that besides a working connection to the controller also a working API is available in the background. If the latter is temporarily unavailable for some reason, a corresponding message is displayed:



This display is visible until the API is available again. If this is the case, a [LOGIN window](#) is displayed and the user is prompted to log in from the new.

9.7 Disclaimer

Products, which are delivered with the operating system versions

Salamander 2 ARM + Salamander 2 x86	≥ Version 09.02.200
-------------------------------------	---------------------

Salamander 3	≥ Version 09.04.060
--------------	---------------------

Gecko ARM + Gecko x86	≥ Version 09.07.100
-----------------------	---------------------

or greater are shipped with an autoexec.lsl file with the WebConfig interface active by default, and the passwords initialized with a default value to facilitate quick configuration.

The WebConfig function **must be deactivated** in the autoexec.lsl file **or the passwords must be adapted** in the following situations:

- before delivery of the controller to an end customer
- before any other use in productive operation (both in-house and at end customers or third parties)
- before connecting the device to a network infrastructure where access by third parties from outside is possible

The SIGMATEK Device Configuration function can be started with the two commands

```
webconfig port 1980
webconfig start
```

These two commands **must be commented out or deleted from the autoexec.lsl file** in order to switch off the SIGMATEK Device Configuration function.

SIGMATEK accepts no liability whatsoever for direct and/or indirect damage or for any claims that may result from or be related to negligent or deliberate non-compliance with this instruction. This also applies to claims made by end customers or third parties in connection with the WebConfig function.

Should SIGMATEK itself suffer any damage as a result of non-compliance with this instruction, or should SIGMATEK be held liable as a result thereof, the business partner shall fully indemnify and **hold SIGMATEK harmless against all damages and claims**. Whether or not the business partner is at fault in the non-observance of the instruction shall be irrelevant in this respect.