

ExecL

COLLABORATORS

	<i>TITLE :</i> ExecL		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	ExecL	1
1.1	ExecL	1
1.2	TMP:Modula-2/ExecL.def	2

Chapter 1

ExecL

1.1 ExecL

Prozeduren

AbortIO	AddDevice	AddHead
AddIntServer	AddLibrary	AddMemList
AddPort	AddResource	AddSemaphore
AddTail	AddTask	Alert
AllocAbs	AllocEntry	AllocMem
AllocSignal	AllocTrap	AllocVec
Allocate	AttemptSemaphore	↔
AttemptSemaphoreShared		
AvailMem	CacheClearE	CacheClearU
CacheControl	CachePostDMA	CachePreDMA
Cause	CheckIO	CloseDevice
CloseLibrary	ColdReboot	CopyMem
CopyMemQuick	CreateIORequest	CreateMsgPort
Deallocate	Debug	DeleteIORequest
DeleteMsgPort	Disable	DoIO
Enable	Enqueue	FindName
FindPort	FindResident	FindSemaphore
FindTask	Forbid	FreeEntry
FreeMem	FreeSignal	FreeTrap
FreeVec	GetCC	GetMsg
InitCode	InitResident	InitSemaphore
InitStruct	Insert	MakeFunctions
MakeLibrary	ObtainSemaphore	↔
ObtainSemaphoreList		
ObtainSemaphoreShared	OldOpenLibrary	OpenDevice
OpenLibrary	OpenResource	Permit
Procure	PutMsg	RawDoFmt
ReleaseSemaphore	ReleaseSemaphoreList	RemDevice
RemHead	RemIntServer	RemLibrary
RemPort	RemResource	RemSemaphore
RemTail	RemTask	Remove
ReplyMsg	SendIO	SetExcept
SetFunction	SetIntVector	SetSR
SetSignal	SetTaskPri	Signal
StackSwap	SumKickData	SumLibrary
SuperState	Supervisor	TypeOfMem

UserState	Vacate	Wait
WaitIO	WaitPort	

Konstanten

execMinVersion	execName
----------------	----------

Variablen

execBase	execVersion
----------	-------------

1.2 TMP:Modula-2/ExecL.def

```

DEFINITION MODULE ExecL {"exec.library", 33};
(* 10-Mar-1992/cn *)

FROM SYSTEM IMPORT ADDRESS, BITSET, BPTR, BYTE, LONGSET, SHORTSET, WORD;

IMPORT
  d:ExecD, R;

VAR
  execVersion:INTEGER;
  execBase:d. ExecBasePtr ;

CONST
  execMinVersion=33;
  execName="exec.library";

PROCEDURE  AbortIO (ioRequest{R.A1}:ADDRESS); CODE -480;

PROCEDURE  AddDevice (device{R.A1}:d. DevicePtr ); CODE -432;

PROCEDURE  AddHead (
  list{R.A0}:d. ListPtr ;
  node{R.A1}:ADDRESS); CODE -240;

PROCEDURE  AddIntServer (
  intNum{R.D0}: LONGINT;
  interrupt{R.A1}:d. InterruptPtr ); CODE -168;

PROCEDURE  AddLibrary (library{R.A1}:d. LibraryPtr ); CODE -396;

PROCEDURE  AddMemList (
  size{R.D0}:LONGINT;
  attributes{R.D1}:d. MemReqSet ;
  pri{R.D2}:LONGINT;
  base{R.A0}:ADDRESS;
  name{R.A1}:ADDRESS):LONGINT; CODE -618;

PROCEDURE  AddPort (port{R.A1}:d. MsgPortPtr ); CODE -354;

PROCEDURE  AddResource (resource{R.A1}:ADDRESS); CODE -486;

PROCEDURE (*36*) AddSemaphore (

```

```
                signalSemaphore{R.A1}:d. SignalSemaphorePtr ); CODE -600;
(*
  genaugenommen gibt es AddSemaphore schon länger, war aber vor der
  Version 36 der exec.library falsch implementiert. Siehe dazu die
  Beschreibung der Prozedur in der 3. Ausgabe des RKM Autodocs&Includes
  auf S. 131.
*)

PROCEDURE AddTail (
    list{R.A0}:d. ListPtr ;
    node{R.A1}:ADDRESS); CODE -246;

PROCEDURE AddTask (
    task{R.A1}:d. TaskPtr ;
    initialPC{R.A2}:ADDRESS;
    finalPC{R.A3}:ADDRESS); CODE -282;

PROCEDURE Alert (
    alertNum{7}:LONGINT;
    parameters{13}:ADDRESS); CODE -108;
(*
  Der zweite Parameter wird ab WB2.0 ignoriert.
*)

PROCEDURE AllocAbs (
    byteSize{R.D0}:LONGINT;
    location{R.A1}:ADDRESS):ADDRESS; CODE -204;

PROCEDURE Allocate (
    freeList{R.A0}:d. MemHeaderPtr ;
    byteSize{R.D0}:LONGINT):ADDRESS; CODE -186;

PROCEDURE AllocEntry (
    memList{R.A0}:d. MemListPtr ):d. MemListPtr ; CODE -222;

PROCEDURE AllocMem (
    byteSize{R.D0}:LONGINT;
    requirements{R.D1}:d. MemReqSet ):ADDRESS; CODE -198;

PROCEDURE AllocSignal (
    signalNum{R.D0}:LONGINT):LONGINT; CODE -330;

PROCEDURE AllocTrap (trapNum{R.D0}:LONGINT):LONGINT; CODE -342;

PROCEDURE(*36*) AllocVec (
    byteSize{R.D0}:LONGINT;
    requirements{R.D1}:d. MemReqSet ):ADDRESS; CODE -684;

PROCEDURE AttemptSemaphore (
    signalSemaphore{R.A0}:d. SignalSemaphorePtr
):BOOLEAN; CODE -576;

PROCEDURE(*37*) AttemptSemaphoreShared(
    signalSemaphore{R.A0}:d. SignalSemaphorePtr
):BOOLEAN; CODE -720;

PROCEDURE AvailMem (
```

```
requirements{R.D1}:d. MemReqSet ):LONGINT; CODE -216;

PROCEDURE (*37*) CacheClearE (
    address{R.A0}:ADDRESS;
    length{R.D0}:LONGCARD;
    caches{R.D1}:d. CacheFlagSet ); CODE -642;

PROCEDURE (*37*) CacheClearU ; CODE -636;

PROCEDURE (*37*) CacheControl (
    cacheBits{R.D0},
    cacheMask{R.D1}:d. CacheFlagSet
):d. CacheFlagSet ; CODE -648;

PROCEDURE (*37*) CachePostDMA (
    vAddress{R.A0}:ADDRESS;
    VAR length{R.A1}:LONGCARD;
    flags{R.D1}:d. DMAControlFlagSet ); CODE -768;

PROCEDURE (*37*) CachePreDMA (
    vAddress{R.A0}:ADDRESS;
    VAR length{R.A1}:LONGCARD;
    flags{R.D1}:d. DMAControlFlagSet ):ADDRESS; CODE -762;

PROCEDURE Cause (interrupt{R.A1}:d. InterruptPtr ); CODE -180;

PROCEDURE CheckIO (ioRequest{R.A1}:ADDRESS):BOOLEAN; CODE -468;

PROCEDURE CloseDevice (ioRequest{R.A1}:ADDRESS); CODE -450;

PROCEDURE CloseLibrary (library{R.A1}:d. LibraryPtr ); CODE -414;

PROCEDURE (*36*) ColdReboot ; CODE -726;

PROCEDURE CopyMem (
    source{R.A0}:ADDRESS;
    dest{R.A1}:ADDRESS;
    size{R.D0}:LONGINT); CODE -624;

PROCEDURE CopyMemQuick (
    source{R.A0}:ADDRESS;
    dest{R.A1}:ADDRESS;
    size{R.D0}:LONGINT); CODE -630;

PROCEDURE (*36*) CreateIORequest (
    replyPort{R.A0}:d. MsgPortPtr ;
    size{R.D0}:LONGINT):ADDRESS ; CODE -654;

PROCEDURE (*36*) CreateMsgPort ():d. MsgPortPtr ; CODE -666;

PROCEDURE Deallocate (
    freeList{R.A0}:d. MemHeaderPtr ;
    memoryBlock{R.A1}:ADDRESS;
    byteSize{R.D0}:LONGINT); CODE -192;

PROCEDURE Debug (flags{R.D0}:LONGSET); CODE -114;
(*
```

Die flags müssen immer den Wert LONGSET{} enthalten.
*)

PROCEDURE(*36*) DeleteIORequest (iorequest{R.A0}:ADDRESS); CODE -660;

PROCEDURE(*36*) DeleteMsgPort (port{R.A0}:d. MsgPortPtr); CODE -672;

PROCEDURE Disable (); CODE -120;

PROCEDURE DoIO (ioRequest{R.A1}:ADDRESS); CODE -456;

PROCEDURE Enable (); CODE -126;

PROCEDURE Enqueue (
list{R.A0}:d. ListPtr ;
node{R.A1}:ADDRESS); CODE -270;

PROCEDURE FindName (
start{R.A0}:ADDRESS;
name{R.A1}:ADDRESS):ADDRESS; CODE -276;

PROCEDURE FindPort (name{R.A1}:ADDRESS):d. MsgPortPtr ; CODE -390;

PROCEDURE FindResident (
name{R.A1}:ADDRESS):d. ResidentPtr ; CODE -96;

PROCEDURE FindSemaphore (
name{R.A1}:ADDRESS):d. SignalSemaphorePtr ; CODE -594;

PROCEDURE FindTask (name{R.A1}:ADDRESS):d. TaskPtr ; CODE -294;

PROCEDURE Forbid (); CODE -132;

PROCEDURE FreeEntry (memList{R.A0}:d. MemListPtr); CODE -228;

PROCEDURE FreeMem (
memoryBlock{R.A1}:ADDRESS;
byteSize{R.D0}:LONGINT); CODE -210;

PROCEDURE FreeSignal (signalNum{R.D0}:LONGINT); CODE -336;

PROCEDURE FreeTrap (trapNum{R.D0}:LONGINT); CODE -348;

PROCEDURE(*36*) FreeVec (memoryBlock{R.A1}:ADDRESS); CODE -690;

PROCEDURE GetCC ():BITSET; CODE -528;

PROCEDURE GetMsg (port{R.A0}:d. MsgPortPtr):ADDRESS; CODE -372;

PROCEDURE InitCode (
startClass{R.D0}:d. ResidentFlagSet ;
version{R.D1}:LONGINT); CODE -72;

PROCEDURE InitResident (
resident{R.A1}:d. ResidentPtr ;
segList{R.D1}:BPTR); CODE -102;

```
PROCEDURE InitSemaphore (
    signalSemaphore{R.A0}:d. SignalSemaphorePtr ); CODE -558;

PROCEDURE InitStruct (
    initTable{R.A1}:ADDRESS;
    memory{R.A2}:ADDRESS;
    size{R.D0}:LONGCARD); CODE -78;

PROCEDURE Insert (
    list{R.A0}:d. ListPtr ;
    node{R.A1}:ADDRESS;
    listNode{R.A2}:ADDRESS); CODE -234;

PROCEDURE MakeFunctions (
    target{R.A0}:ADDRESS;
    funcArray{R.A1}:ADDRESS;
    funcDispBase{R.A2}:ADDRESS); CODE -90;

PROCEDURE MakeLibrary (
    vectors{R.A0}:ADDRESS;
    structure{R.A1}:ADDRESS;
    init{R.A2}:ADDRESS;
    dataSize{R.D0}:LONGINT;
    segList{R.D1}:BPTR):d. LibraryPtr ; CODE -84;

PROCEDURE ObtainSemaphore (
    signalSemaphore{R.A0}:d. SignalSemaphorePtr ); CODE -564;

PROCEDURE ObtainSemaphoreList (list{R.A0}:d. ListPtr ); CODE -582;

PROCEDURE(*36*) ObtainSemaphoreShared (
    sigSem{R.A0}:d. SignalSemaphorePtr ); CODE -678;

PROCEDURE OldOpenLibrary (
    libName{R.A1}:ADDRESS):d. LibraryPtr ; CODE -408;

PROCEDURE OpenDevice (
    devName{R.A0}:ADDRESS;
    unitNumber{R.D0}:LONGINT;
    ioRequest{R.A1}:ADDRESS;
    flags{R.D1}:LONGSET); CODE -444;

PROCEDURE OpenLibrary (
    libName{R.A1}:ADDRESS;
    version{R.D0}:LONGINT):d. LibraryPtr ; CODE -552;

PROCEDURE OpenResource (resName{R.A1}:ADDRESS):ADDRESS; CODE -498;

PROCEDURE Permit (); CODE -138;

PROCEDURE Procure (
    semaphore{R.A0}:d. SemaphorePtr ;
    bidMessage{R.A1}:d. MessagePtr ):LONGINT; CODE -540;

PROCEDURE PutMsg (
    port{R.A0}:d. MsgPortPtr ;
    message{R.A1}:ADDRESS); CODE -366;
```

```
PROCEDURE RawDoFmt (
    formatString{R.A0}:ADDRESS;
    dataStream{R.A1}:ADDRESS;
    putChProc{R.A2}:ADDRESS;
    putChData{R.A3}:ADDRESS); CODE -522;

PROCEDURE ReleaseSemaphore (
    signalSemaphore{R.A0}:d. SignalSemaphorePtr ); CODE -570;

PROCEDURE ReleaseSemaphoreList (list{R.A0}:d. ListPtr ); CODE -588;

PROCEDURE RemDevice (device{R.A1}:d. DevicePtr ); CODE -438;

PROCEDURE RemHead (list{R.A0}:d. ListPtr ):ADDRESS; CODE -258;

PROCEDURE RemIntServer (
    intNum{R.D0}: LONGINT;
    interrupt{R.A1}:d. InterruptPtr ); CODE -174;

PROCEDURE RemLibrary (
    library{R.A1}:d. LibraryPtr ); CODE -402;

PROCEDURE Remove (node{R.A1}:ADDRESS); CODE -252;

PROCEDURE RemPort (port{R.A1}:d. MsgPortPtr ); CODE -360;

PROCEDURE RemResource (resource{R.A1}:ADDRESS); CODE -492;

PROCEDURE RemSemaphore (
    signalSemaphore{R.A1}:d. SignalSemaphorePtr ); CODE -606;

PROCEDURE RemTail (list{R.A0}:d. ListPtr ):ADDRESS; CODE -264;

PROCEDURE RemTask (task{R.A1}:d. TaskPtr ); CODE -288;

PROCEDURE ReplyMsg (message{R.A1}:ADDRESS); CODE -378;

PROCEDURE SendIO (ioRequest{R.A1}:ADDRESS); CODE -462;

PROCEDURE SetExcept (
    newSignals{R.D0}:LONGSET;
    signalMask{R.D1}:LONGSET):LONGSET; CODE -312;

PROCEDURE SetFunction (
    library{R.A1}:d. LibraryPtr ;
    funcOffset{R.A0}:INTEGER;
    funcEntry{R.D0}:ADDRESS):ADDRESS; CODE -420;

PROCEDURE SetIntVector (
    intNumber{R.D0}:LONGINT;
    interrupt{R.A1}:d. InterruptPtr ):d. InterruptPtr ; CODE -162;

PROCEDURE SetSignal (
    newSignals{R.D0}:LONGSET;
    signalMask{R.D1}:LONGSET):LONGSET; CODE -306;
```

```
PROCEDURE SetSR (
    newSR{R.D0}:BITSET;
    mask{R.D1}:BITSET):BITSET; CODE -144;

PROCEDURE SetTaskPri (
    task{R.A1}:d. TaskPtr ;
    priority{R.D0}:SHORTINT):SHORTINT; CODE -300;

PROCEDURE Signal (
    task{R.A1}:d. TaskPtr ;
    signals{R.D0}:LONGSET); CODE -324;

PROCEDURE(*37*) StackSwap (
    VAR newStack{R.A0}:d. StackSwapStruct ); CODE -732;

PROCEDURE SumKickData ():LONGCARD; CODE -612;
(*
  Das Resultat ist undefiniert bei WB1.2 oder WB1.3.
*)

PROCEDURE SumLibrary (library{R.A1}:d. LibraryPtr ); CODE -426;

PROCEDURE SuperState ():ADDRESS; CODE -150;

PROCEDURE Supervisor (userFunction{R.A5}:PROC); CODE -30;

PROCEDURE TypeOfMem (address{R.A1}:ADDRESS):d. MemReqSet ; CODE -534;

PROCEDURE UserState (sysStack{R.D0}:ADDRESS); CODE -156;

PROCEDURE Vacate (semaphore{R.A0}:d. SemaphorePtr ); CODE -546;

PROCEDURE Wait (signalSet{R.D0}:LONGSET):LONGSET; CODE -318;

PROCEDURE WaitIO (ioRequest{R.A1}:ADDRESS); CODE -474;

PROCEDURE WaitPort (port{R.A0}:d. MsgPortPtr ); CODE -384;

END ExecL.lib33
```
