

ExecD

COLLABORATORS

	<i>TITLE :</i> ExecD		
<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	ExecD	1
1.1	ExecD	1
1.2	TMP:Modula-2/ExecD.def	3

Chapter 1

ExecD

1.1 ExecD

Konstanten

abortIO	aborted	addSWGadget
asyncPkt	audioDev	audioDevObj
badAddress	badChkSum	badExpansionFree
badFreeAddr	badGadget	badInitFunc
badLength	badMessage	badOverlay
badParm	badSegList	badState
base	baseChkSum	beginIO
bitMap	bltBitMap	bogusExcpt
bootError	bootStrap	bootStrapObj
ciaRsrc	ciaRsrcObj	clear
close	closeDev	closeLib
consoleDev	consoleDevObj	createPort
deadEnd	diskBlkSeq	diskCopy
diskCopyObj	diskError	diskRsrc
diskRsrcObj	diskfontLib	diskfontLibObj
dosLib	dosLibObj	drHasDisk
drIntNoAct	endTask	excptVect
execLib	execLibObj	execName
expansionLib	expansionLibObj	expunge
extFunc	fileReclosed	flush
freeTwice	freeVec	gadTools
gadToolsObj	gadgetType	gamePortDev
gamePortDevObj	gfxFreeError	gfxNewError
gfxNoLCM	gfxNoMem	gfxNoMemMspc
graphicsLib	graphicsLibObj	iconLib
iconLibObj	initAPtr	intrMem
intuition	intuitionObj	invalid
ioAfterClose	ioError	ioUsedTwice
itemAlloc	itemBoxTop	keyFree
keyRange	keyboardDev	keyboardDevObj
layersLib	layersLibObj	layersNoMem
libChkSum	longFrame	makeLib
makeVPort	matchword	mathLib
mathLibObj	memCorrupt	memoryInsane
miscRsrc	miscRsrcObj	never
nmi	noCmd	noConsole

noFonts	noMemory	noSignal
noWindow	nonStd	nonstd
obsoleteFont	open	openDev
openFail	openLib	openRes
openScreen	openScrnRast	openWindow
planeAlloc	procCreate	qPktFail
quick	ramLib	ramLibObj
read	recovery	regionMemory
reserved	reset	selfTest
semCorrupt	shortFrame	sigAbort
sigBlit	sigChild	sigDos
sigIntuition	sigSingle	stackProbe
start	startMem	stop
subAlloc	sysScrnType	tdCalibSeek
tdDelay	textTmpRas	timerDev
timerDevObj	tmBadReq	tmBadSupply
trackDiskDev	trackDiskDevObj	unitBusy
unknownObj	unknownSubsystem	update
userDef	utilityLib	utilityLibObj
vectSize	wbAddToolMenuItem	wbBadIOMsg
wbBadStartupMsg1	wbBadStartupMsg2	←
wbCreateWBMenuCreateMenus1		
wbCreateWBMenuCreateMenus2	wbInitLayerDemon	←
wbInitPotionAllocDrawer		
wbInitScreenAndWindows1	wbInitScreenAndWindows2	←
wbInitScreenAndWindows3		
wbInitTimer	wbInitWbGels	←
wbLayoutWBMenuLayoutMenus		
wbMalloc	wbReLayoutToolMenu	weirdEcho
workbench	workbenchObj	write

Variablen

execBase

Typ-Deklarationen

AttnFlagSet	AttnFlags	CacheFlagSet
CacheFlags	DMAControlFlagSet	DMAControlFlags
Device	DevicePtr	ExecBase
ExecBasePtr	IOFlagSet	IORequest
IORequestPtr	IOWStdReq	IOWStdReqPtr
IntFlags	IntVector	Interrupt
InterruptPtr	LibFlagSet	LibFlags
Library	LibraryPtr	List
ListPtr	MemChunk	MemChunkPtr
MemEntry	MemHeader	MemHeaderPtr
MemList	MemListPtr	MemReqSet
MemReqs	MemTypeSet	Message
MessagePtr	MinList	MinListPtr
MinNode	MinNodePtr	MsgPort
MsgPortAction	MsgPortPtr	Node
NodePtr	NodeType	Resident
ResidentFlagSet	ResidentFlags	ResidentPtr
Semaphore	SemaphorePtr	SemaphoreRequest
SignalSemaphore	SignalSemaphorePtr	SoftIntList

StackSwapStruct	Task	TaskFlagSet
TaskFlags	TaskPtr	TaskState
Unit	UnitFlagSet	UnitFlags
UnitPtr		

1.2 TMP:Modula-2/ExecD.def

```

DEFINITION MODULE ExecD; (*$ Implementation:=FALSE *)
(* 02-Feb-1992/cn *)

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

CONST
  execName="exec.library";

TYPE
  (*
    Listenverwaltung
  *)
  NodeType = (
    unknown, task, interrupt, device, msgPort, message, freeMsg,
    replyMsg, resource, library, memory, softInt, font, process,
    semaphore, signalSem, bootNode, kickMem, graphics, deathMessage
  );
  NodePtr = POINTER TO Node ;
  Node = RECORD
    succ: NodePtr ;
    pred: NodePtr ;
    type: NodeType ;
    pri: SHORTINT;
    name: ADDRESS;
  END;
  MinNodePtr = POINTER TO MinNode ;
  MinNode = RECORD
    succ: MinNodePtr ;
    pred: MinNodePtr ;
  END;
  List = RECORD
    head: NodePtr ;
    tail: NodePtr ;
    tailPred: NodePtr ;
    type: NodeType ;
    pad: BYTE;
  END;
  ListPtr = POINTER TO List ;
  MinList = RECORD
    head: MinNodePtr ;
    tail: MinNodePtr ;
    tailPred: MinNodePtr ;
  END;
  MinListPtr = POINTER TO MinList ;
  (*
    Interrupts
  *)
  IntFlags = (

```

```

    theInt,dskblk,ifSoftint,ports,coper,vertb,blit,aud0i,
    aud1i,aud2i,aud3i,rbfInt,disksync,exter,inten,intSet
);
Interrupt =RECORD
    node: Node ;
    data:ADDRESS;
    code:PROC;
END;
InterruptPtr =POINTER TO Interrupt ;
IntVector =RECORD
    data:ADDRESS;
    code:PROC;
    node: NodePtr ;
END;
SoftIntList =RECORD
    list: List ;
    pad:WORD;
END;

CONST
    nmi=intSet;

(*
    Speicherverwaltung
*)
TYPE
    MemReqs =(
        public,chip,fast,mr3,mr4,mr5,mr6,mr7,
        local,dma24Bit,mr10,mr11,mr12,mr13,mr14,mr15,memClear,largest,
        reverse,total
    );
(*
    Beachte:
    "MemReqSet" ist 32 bit gross und enthält alle Flags aus "MemReqs".
    "MemTypeSet" ist nur 16 bit gross und enthält die von gewissen
    Speicherverwaltungsprozeduren zusätzlich benötigten Flags
    "memClear", "largest", "reverse" und "total" nicht.
*)
    MemReqSet =SET OF MemReqs ;
    MemTypeSet =SET OF [public..mr15];
    MemChunkPtr =POINTER TO MemChunk ;
    MemChunk =RECORD
        next: MemChunkPtr ;
        bytes:LONGCARD;
END;
    MemHeader =RECORD
        node: Node ;
        attributes: MemTypeSet ;
        first: MemChunkPtr ;
        lower:ADDRESS;
        upper:ADDRESS;
        free:LONGCARD;
END;
    MemHeaderPtr =POINTER TO MemHeader ;
    MemEntry =RECORD
        CASE :INTEGER OF
            | 1: reqs: MemReqSet

```

```

| 2: addr:ADDRESS
END;
length:LONGCARD;
END;
MemList =RECORD
node: Node ;
numEntries:CARDINAL;
(*
der hier anschliessende Speicherplatz enthält MemEntry
Elemente. Am besten deklariert man sich je nach Fall einen
eignen Typ, der aus einem MemList Feld gefolgt von einem
MemEntry Array. Das Beispiel auf Seite 135 der 3. Ausgabe
des RKM Autodocs & Includes würde in M2Amiga so aussehen:

```

```

VAR
mem=RECORD
list:MemList;
entries:ARRAY [1..5] OF MemEntry;
END;
res:MemListPtr;
BEGIN
mem.list.numEntries:=5;
mem.entries[1].reqs:=MemReqSet{memClear};
mem.entries[1].length:=2;
mem.entries[2].reqs:=MemReqSet{public};
mem.entries[2].length:=4;
mem.entries[3].reqs:=MemReqSet{chip,memClear};
mem.entries[3].length:=8;
mem.entries[4].reqs:=MemReqSet{memClear};
mem.entries[4].length:=16;
mem.entries[5].reqs:=MemReqSet{public,memClear};
mem.entries[5].length:=32;
res:=AllocEntry(ADR(mem));
IF LONGINT(ADDRESS(res))>0 THEN
(* Speicherallozierung war erfolgreich. *)
*)
END;
MemListPtr =POINTER TO MemList ;

(*
Tasks
*)
TaskFlags =(
procTime,tfl,tfl2,eTask,stackChk,exception,switch,launch
);
TaskFlagSet =SET OF TaskFlags ;
TaskState =(inval,added,run,ready,wait,except,removed);
Task =RECORD
node: Node ;
flags: TaskFlagSet ;
state: TaskState ;
idNestCnt:SHORTINT;
tdNestCnt:SHORTINT;
sigAlloc:LONGSET;
sigWait:LONGSET;
sigRecvd:LONGSET;
sigExcept:LONGSET;

```

```
    trapAlloc:BITSET;
    trapAble:BITSET;
    exceptData:ADDRESS;
    exceptCode:PROC;
    trapData:ADDRESS;
    trapCode:PROC;
    spReg:ADDRESS;
    spLower:ADDRESS;
    spUpper:ADDRESS;
    switch:PROC;
    launch:PROC;
    memEntry: List ;
    userData:ADDRESS;
END;
TaskPtr =POINTER TO Task ;

StackSwapStruct =RECORD
    lower:ADDRESS;
    upper:ADDRESS;
    pointer:ADDRESS;
END;

CONST
(*
Vordefinierte Signalnummern
*)
    sigAbort=0;
    sigChild=1;
    sigBlit=4;
    sigSingle=sigBlit;
    sigIntuition=5;
    sigDos=8;

(*
Alertarten
*)
    deadEnd=0800000000H;
    recovery=0000000000H;
(*
Allgemeine Alertcodes
*)
    noMemory=000010000H;
    makeLib=000020000H;
    openLib=000030000H;
    openDev=000040000H;
    openRes=000050000H;
    ioError=000060000H;
    noSignal=000070000H;
    badParm=000080000H;
    closeLib=000090000H;
    closeDev=0000A0000H;
    procCreate=0000B0000H;
(*
Alertobjekte
*)
    execLibObj=000008001H;
    graphicsLibObj=000008002H;
```

```
layersLibObj=000008003H;
intuitionObj=000008004H;
mathLibObj=000008005H;
dosLibObj=000008007H;
ramLibObj=000008008H;
iconLibObj=000008009H;
expansionLibObj=00000800AH;
diskfontLibObj=00000800BH;
utilityLibObj=00000800CH;
audioDevObj=000008010H;
consoleDevObj=000008011H;
gamePortDevObj=000008012H;
keyboardDevObj=000008013H;
trackDiskDevObj=000008014H;
timerDevObj=000008015H;
ciaRsrcObj=000008020H;
diskRsrcObj=000008021H;
miscRsrcObj=000008022H;
bootStrapObj=000008030H;
workbenchObj=000008031H;
diskCopyObj=000008032H;
gadToolsObj=000008033H;
unknownObj=000008035H;
(*
Spezifische Alerts
*)
execLib=001000000H;
excptVect=001000001H;
baseChkSum=001000002H;
libChkSum=001000003H;
memCorrupt=081000005H;
intrMem=081000006H;
initAPtr=001000007H;
semCorrupt=001000008H;
freeTwice=001000009H;
bogusExcpt=08100000AH;
ioUsedTwice=00100000BH;
memoryInsane=00100000CH;
ioAfterClose=00100000DH;
stackProbe=00100000EH;
badFreeAddr=00100000FH;
graphicsLib=002000000H;
gfxNoMem=082010000H;
gfxNoMemMspc=082010001H;
longFrame=082010006H;
shortFrame=082010007H;
textTmpRas=002010009H;
bltBitMap=08201000AH;
regionMemory=08201000BH;
makeVPort=082010030H;
gfxNewError=00200000CH;
gfxFreeError=00200000DH;
gfxNoLCM=082011234H;
obsoleteFont=002000401H;
layersLib=003000000H;
layersNoMem=083010000H;
intuition=004000000H;
```

```
gadgetType=0840000001H;
badGadget=0040000001H;
createPort=0840100002H;
itemAlloc=0040100003H;
subAlloc=0040100004H;
planeAlloc=0840100005H;
itemBoxTop=0840000006H;
openScreen=0840100007H;
openScrnRast=0840100008H;
sysScrnType=0840000009H;
addSWGadget=084010000AH;
openWindow=084010000BH;
badState=084000000CH;
badMessage=084000000DH;
weirdEcho=084000000EH;
noConsole=084000000FH;
mathLib=0050000000H;
dosLib=0070000000H;
startMem=0070100001H;
endTask=0070000002H;
qPktFail=0070000003H;
asyncPkt=0070000004H;
freeVec=0070000005H;
diskBlkSeq=0070000006H;
bitMap=0070000007H;
keyFree=0070000008H;
badChkSum=0070000009H;
diskError=007000000AH;
keyRange=007000000BH;
badOverlay=007000000CH;
badInitFunc=007000000DH;
fileReclosed=007000000EH;
ramLib=0080000000H;
badSegList=0080000001H;
iconLib=0090000000H;
expansionLib=00A0000000H;
badExpansionFree=00A0000001H;
diskfontLib=00B0000000H;
audioDev=0100000000H;
consoleDev=0110000000H;
noWindow=0110000001H;
gamePortDev=0120000000H;
keyboardDev=0130000000H;
trackDiskDev=0140000000H;
tdCalibSeek=0140000001H;
tdDelay=0140000002H;
timerDev=0150000000H;
tmBadReq=0150000001H;
tmBadSupply=0150000002H;
ciaRsrc=0200000000H;
diskRsrc=0210000000H;
drHasDisk=0210000001H;
drIntNoAct=0210000002H;
miscRsrc=0220000000H;
bootStrap=0300000000H;
bootError=0300000001H;
workbench=0310000000H;
```

```

noFonts=0B10000001H;
wbBadStartupMsg1=0310000001H;
wbBadStartupMsg2=0310000002H;
wbBadIOMsg=0310000003H;
wbInitPotionAllocDrawer=0B10100004H;
wbCreateWBMenuCreateMenus1=0B10100005H;
wbCreateWBMenuCreateMenus2=0B10100006H;
wbLayoutWBMenuLayoutMenus=0B10100007H;
wbAddToolMenuItem=0B10100008H;
wbReLayoutToolMenu=0B10100009H;
wbInitTimer=0B1010000AH;
wbInitLayerDemon=0B1010000BH;
wbInitWbGels=0B1010000CH;
wbInitScreenAndWindows1=0B1010000DH;
wbInitScreenAndWindows2=0B1010000EH;
wbInitScreenAndWindows3=0B1010000FH;
wbMAlloc=0B10100010H;
diskCopy=0320000000H;
gadTools=0330000000H;
utilityLib=0340000000H;
unknownSubsystem=0350000000H;

```

```

(*
Meldungen
*)
TYPE
    MsgPortAction =(signal, softint, ignore);
    MsgPort =RECORD
        node: Node ;
        CASE flags: MsgPortAction OF
            | signal:
                sigBit:SHORTCARD;
                sigTask: TaskPtr ;
            | softint:
                pad0:BYTE;
                softInt: InterruptPtr ;
            | ignore:
                pad1:BYTE;
                pad2:ADDRESS
        END;
        msgList: List ;
    END;
    MsgPortPtr =POINTER TO MsgPort ;
    Message =RECORD
        node: Node ;
        replyPort: MsgPortPtr ;
        length:CARDINAL;
    END;
    MessagePtr =POINTER TO Message ;

(*
Bibliotheken
*)
CONST
    vectSize=6;
    reserved=4;
    base=-vectSize;

```

```

userDef=base-reserved*vectSize;
nonStd=userDef;
open=-6;
close=-12;
expunge=-18;
extFunc=-24;

```

```
TYPE
```

```

    LibFlags =(summing,changed,sumUsed,delExp);
    LibFlagSet =SET OF  LibFlags ;
    Library =RECORD
        node: Node ;
        flags: LibFlagSet ;
        pad:BYTE;
        negSize:CARDINAL;
        posSize:CARDINAL;
        version:CARDINAL;
        revision:CARDINAL;
        idString:ADDRESS;
        sum:LONGCARD;
        openCnt:CARDINAL;
    END;
    LibraryPtr =POINTER TO  Library ;

```

```
(*
```

```
Devices und I/O
```

```
*)
```

```

    Device =RECORD
        library: Library ;
    END;
    DevicePtr =POINTER TO  Device ;
    UnitFlags =(active,inTask,uf2,uf3,uf4,uf5,uf6,uf7);
    UnitFlagSet =SET OF  UnitFlags ;
    Unit =RECORD
        msgPort: MsgPort ;
        flags: UnitFlagSet ;
        pad:BYTE;
        openCnt:CARDINAL;
    END;
    UnitPtr =POINTER TO  Unit ;

```

```
TYPE
```

```
    IOFlagSet =SHORTSET;
```

```
CONST
```

```
    quick= IOFlagSet {0};
```

```
TYPE
```

```

    IORequest =RECORD
        message: Message ;
        device: DevicePtr ;
        unit: UnitPtr ;
        command:CARDINAL;
        flags: IOFlagSet ;
        error:SHORTINT;
    END;
    IORequestPtr =POINTER TO  IORequest ;

```

```
IOStdReq =RECORD
  message: Message ;
  device: DevicePtr ;
  unit: UnitPtr ;
  command: CARDINAL;
  flags: IOFlagSet ;
  error: SHORTINT;
  actual: LONGCARD;
  length: LONGINT;
  data: ADDRESS;
  offset: LONGCARD;
END;
IOStdReqPtr =POINTER TO IOStdReq ;

CONST
(*
  Offsets der devicespezifischen Funktionen
*)
  abortIO=-36;
  beginIO=-30;
(*
  Standardbefehle für IORequest.command
*)
  invalid=0;
  reset=1;
  read=2;
  write=3;
  update=4;
  clear=5;
  stop=6;
  start=7;
  flush=8;
  nonstd=9;
(*
  Standard Fehlerwerte fuer IORequest.error
*)
  openFail=-1;
  aborted=-2;
  noCmd=-3;
  badLength=-4;
  badAddress=-5;
  unitBusy=-6;
  selfTest=-7;

(*
  Semaphore
*)
TYPE
  Semaphore =RECORD
    msgPort: MsgPort ;
    bids: INTEGER;
  END;
  SemaphorePtr =POINTER TO Semaphore ;
  SemaphoreRequest =RECORD
    link: MinNode ;
    waiter: TaskPtr ;
  END;
```

```

SignalSemaphore =RECORD
  link: Node ;
  nestCount:INTEGER;
  waitQueue: MinList ;
  multipleLink: SemaphoreRequest ;
  owner: TaskPtr ;
  queueCount:INTEGER;
END;
SignalSemaphorePtr =POINTER TO  SignalSemaphore ;

(*
  Residentstruktur
*)
TYPE
  ResidentFlags =(
    coldstart,singleTask,afterDos,rf3,rf4,rf5,rf6,autoinit
  );
  ResidentFlagSet =SET OF  ResidentFlags ;
  ResidentPtr =POINTER TO  Resident ;
  Resident =RECORD
    matchWord:CARDINAL;
    matchTag: ResidentPtr ;
    endSkip:ADDRESS;
    flags: ResidentFlagSet ;
    version:SHORTCARD;
    type: NodeType ;
    pri:SHORTINT;
    name:ADDRESS;
    idString:ADDRESS;
    init:ADDRESS;
  END;

CONST
  matchword=04AFCH;
  never= ResidentFlagSet {};

(*
  Librarystruktur vn Exec
*)
TYPE
  AttnFlags =(
    m68010,
    m68020,
    m68030,
    m68040,
    m68881,
    m68882,
    fpu40,
    af7,reserved8,reserved9,af10,af11,af12,af13,af14,private
  );
  AttnFlagSet =SET OF  AttnFlags ;

  CacheFlags =(
    enableI,freezeI,cf2,clearI,ibe,cf5,cf6,cf7,
    enableD,freezeD,cf10,clearD,dbe,writeAllocate,cf14,cf15,
    cf16,cf17,cf18,cf19,cf20,cf21,cf22,cf23,
    cf24,cf25,cf26,cf27,cf28,cf29,cf30,copyBack

```

```

);
CacheFlagSet =SET OF  CacheFlags ;

DMAControlFlags =(continue,noModify);
DMAControlFlagSet =SET OF  DMAControlFlags ;

ExecBase =RECORD
  libNode: Library ;
  softVer: CARDINAL;
  lowMemChkSum: INTEGER;
  chkBase: LONGCARD;
  coldCapture: ADDRESS;
  coolCapture: ADDRESS;
  warmCapture: ADDRESS;
  sysStkUpper: ADDRESS;
  sysStkLower: ADDRESS;
  maxLocMem: LONGCARD;
  debugEntry: ADDRESS;
  debugData: ADDRESS;
  alertData: ADDRESS;
  maxExtMem: ADDRESS;
  chkSum: CARDINAL;
  intVects: ARRAY  IntFlags  OF  IntVector ;
  thisTask: TaskPtr ;
  idleCount: LONGCARD;
  dispCount: LONGCARD;
  quantum: CARDINAL;
  elapsed: CARDINAL;
  sysFlags: CARDINAL;
  idNestCnt: SHORTINT;
  tdNestCnt: SHORTINT;
  attnFlags: AttnFlagSet ;
  attnResched: CARDINAL;
  resModules: ADDRESS;
  taskTrapCode: PROC;
  taskExceptCode: PROC;
  taskExitCode: PROC;
  taskSigAlloc: LONGSET;
  taskTrapAlloc: BITSET;
  memList: List ;
  resourceList: List ;
  deviceList: List ;
  intrList: List ;
  libList: List ;
  portList: List ;
  taskReady: List ;
  taskWait: List ;
  softInts: ARRAY [0..4] OF  SoftIntList ;
  lastAlert: ARRAY [0..3] OF  LONGINT;
  vBlankFrequency: SHORTCARD;
  powerSupplyFrequency: SHORTCARD;
  semaphoreList: List ;
  kickMemPtr: ADDRESS;
  kickTagPtr: ADDRESS;
  kickChecksum: LONGCARD;
  (*
    Neu ab Version 36:

```

```
*)
pad0: CARDINAL;
launchPoint: LONGCARD;
ramLibPrivate: ADDRESS;
eClockFrequency: LONGCARD;
cacheControl: CacheFlagSet ;
taskId: LONGCARD;
puddleSize: LONGCARD;
poolThreshold: LONGCARD;
publicPool: MinList ;
mmuLock: ADDRESS;
reserved: ARRAY[0..11] OF BYTE;
END;
ExecBasePtr = POINTER TO ExecBase ;

VAR
  execBase[4]: ExecBasePtr ;

END ExecD.noimp
```
