Table of Contents

SDT	rw	/SFB_501/teams/team3/Basis−Projekt.BaX/Entwicklung/RequirementsSp
	rw	/SFB_501/teams/team3/Basis−Projekt.BaX/Entwicklung/RequirementsSp

Used Packages

DataTypes	r–	../Datatypes/LatestRev/DataTypes.sun
Utils	r–	../Utils/LatestRev/Utils.sun

SensorsPackage

Sensors	rw	Sensors.sun
ODLSens	rw	ODLSens.sbt
ODLSens	rw	ODLSens.spr
setPhysVal	rw	setAnalogVal.spd
MotDet	rw	MotDet.sbt
MotDet	rw	MotDet.spr
setPhysVal	rw	setBinaryVal.spd
Switch	rw	Switch.sbt
Switch	rw	Switch.spr
setPhysVal	rw	setBinaryVal1.spd
SunDet	rw	SunDet.sbt
x:y ols1 : ODLSens		
x:y ols2 : ODLSens		
x:y ols3 : ODLSens		
x:y ols4 : ODLSens		
x:y ols5 : ODLSens		
x:y ols6 : ODLSens		
SunDetCtrl	rw	SunDetCtrl.sbk
SunDetCtrl	rw	SunDetCtrl.spr
xi1	rw	xi1.sbk
addId1	rw	addId1.spr
xi2	rw	xi2.sbk
addId2	rw	addId2.spr
xi3	rw	xi3.sbk
addId3	rw	addId3.spr
xi4	rw	xi4.sbk
addId4	rw	addId4.spr
xi5	rw	xi5.sbk

addId5	rw	addId5.spr
xi6	rw	xi6.sbk
addId6	rw	addId6.spr
Panel	rw	Panel.sbk
Panel	rw	Panel.spr
Monitor	rw	Monitor.sbk
Monitor	rw	Monitor.spr

ecification/Sensors/Sensors.sdt

ecification/Sensors/

Package Datatypes


```
SIGNAL dummy;
SIGNAL ident (PID, CharString);
```

```
SYNTYPE Analog=Real  ENDSYNTYPE;
SYNTYPE Binary=Integer CONSTANTS 0,1 ENDSYNTYPE;
SYNTYPE Percent=Real CONSTANTS 0:100 ENDSYNTYPE;
SYNTYPE Day=Integer CONSTANTS 1:365 ENDSYNTYPE;
SYNTYPE WkDay=Integer CONSTANTS 1:7 ENDSYNTYPE;
SYNTYPE Celsius=Real CONSTANTS -60:100 ENDSYNTYPE;
SYNTYPE Lux=REAL CONSTANTS -1:120000 ENDSYNTYPE;
SYNTYPE HallwayIdType = Integer CONSTANTS 1:3 ENDSYNTYPE;
SYNTYPE DegF=Real CONSTANTS -60:200 ENDSYNTYPE; /*Fahrenheit*/
SYNTYPE GPM=Real CONSTANTS 0:200 ENDSYNTYPE; /*Gallon per minute*/
SYNTYPE Psi=Real CONSTANTS 0:200 ENDSYNTYPE; /*Pound per square inch*/
SYNTYPE Watt=Real CONSTANTS 0:10000 ENDSYNTYPE; /*Power unit*/
SYNTYPE ErrorType = INTEGER CONSTANTS 0:100 ENDSYNTYPE;
SYNTYPE Joule = REAL CONSTANTS 0:3000000 ENDSYNTYPE;
SYNTYPE DayTime = INTEGER ENDSYNTYPE;
SYNTYPE LightSourceType = INTEGER CONSTANTS 1:3 ENDSYNTYPE;
```

```
NEWTYPE RoomIdType
struct
rId CHARSTRING;
sId CHARSTRING;
ENDNEWTYPE;
```

```
NEWTYPE LightSourceListType
ARRAY (INTEGER, LightSourceType)
ENDNEWTYPE;
```

```
/* Fehlt: Die Prioritaetsliste fuer die Lichtquellen */
NEWTYPE LightScene
struct
name CharString;
rIll Lux;
dIll Lux;
ls LightSourceListType;
ENDNEWTYPE;
```

```
NEWTYPE CalTimeType
struct
year, month, day, hour, min, sec INTEGER;
operators
   toTime: CalTimeType -> Time;
   fromTime: Time->CalTimeType;
/*#ADT (B)*/
ENDNEWTYPE;
```

Package Datatypes


```
SIGNAL getIlluminance ;
SIGNAL newIlluminance (RoomIdType, Lux);
```

```
SIGNAL newHwEnergyConsumption (Joule, Duration);
SIGNAL newHwMalFunction( Errortype );
SIGNAL getHwPBDelay;
SIGNAL getHwOccDelay;

SIGNAL setHwEnergyDiffTime(Duration);
SIGNAL setHwLight( Boolean );
SIGNAL setHwPbDelay(Duration);
SIGNAL setHwOccDelay (Duration);

SIGNAL setRoomDelay (RoomIdType, Duration);
SIGNAL setRoomReoccDelay (RoomIdType, Duration);
SIGNAL setDefaultLightScene (RoomIdType, LightScene);
SIGNAL setAllDefaultLightScene (LightScene);
SIGNAL setDefaultAmbientLightLevel (RoomIdType, Lux);
SIGNAL setRoomLight (RoomIdType, Boolean);
SIGNAL setRoomEnergyDiffTime (Duration);
SIGNAL newRoomMalfunction (RoomIdType, ErrorType);
SIGNAL newRoomEnergyConsumption (RoomIdType, Joule, Duration);

SIGNAL newEntrance;
```

```
USE DataTypes;
```

Package Utils


```
/*setxxValue: root sets value in remote node*/
/*pollxxValue: root polls value of remote node*/
/*putxxValue: remote node puts value into root*/

REMOTE PROCEDURE setPhysVal; FPAR IN  Analog;
REMOTE PROCEDURE putAnalogVal; FPAR IN  PID, Analog;
REMOTE PROCEDURE putBinaryVal; FPAR IN  PID, Binary;
REMOTE PROCEDURE pollPhysVal; FPAR IN  PID;
```

```
procedure toJoule;
fpar IN w Watt,
IN dimV Percent,
IN timeDiff Duration;
RETURNS Joule;
EXTERNAL;
```

```
/* Proz zum Ausdruck eines beliebigen
Binaerwerts mit der aktuellen Zeit */
procedure printBi;
fpar IN c Charstring,
IN b Binary,
IN dur Time;
EXTERNAL;
```

```
/* Proz zum Ausdruck eines beliebigen
Anaolgwerts mit der aktuellen Zeit */
procedure printAn;
fpar IN c CharString,
IN a Analog,
IN dur Time;
EXTERNAL;
```

```
/* Proz zum Umformen der Kalendereit in die aktuelle Zeit*/
/*procedure calTimeToTime;
fpar IN actTime calTimeType;
RETURNS Time;
EXTERNAL;*/
```

```
/* Proz zum Ausdruck eines beliebigen
Strings mit der aktuellen Zeit */
procedure printIt;
fpar IN c CharString,
IN dur Time;
EXTERNAL;
```

```
/* Proz zum Umformen der aktuellen Zeit in Kalenderzeit*/
/*procedure timeToCalTime;
fpar IN actTime Time;
RETURNS CalTimeType;
EXTERNAL;*/
```

USE DataTypes;

Package Utils


```
/* Proz zum Verbindungsaufbaus des Systems
mit der Umwelt (physSock)*/
procedure initSockets;
EXTERNAL;
```

```
/* Proz., die periodisch einen eventuellen Eingang an dem beiden
Socket physSock bearbeitet.*/
 procedure checkSockets;
returns INTEGER;
EXTERNAL;
```

```
/* Proz., die einen binaeren physik. Werte setzen.*/
procedure setBinOutVal;
FPAR IN actId CharString,
IN val Binary;
EXTERNAL;
```

```
/* Proz., die einen analogen physik. Werte setzen.*/
procedure setAnOutVal;
FPAR IN actId CharString,
IN val Analog;
EXTERNAL;
```

```
/* Proz., die einen physik. Werte anfordert.*/
procedure reqPhysVal;
FPAR IN actId CharString;
EXTERNAL;
```

```
USE Datatypes;
USE Utils;
```

Package Sensors

1(1)

ODLSens

```
/* Signals at object input oi */
SIGNAL getOls;  /*  Get value, independent of mode  */
SIGNAL setOlsPeriod(Duration) ; /* Send value periodically with period p  */
SIGNAL setOlsEvent (Lux); /* Send value if changed  */
SIGNAL setOlsPoll ; /* Wait for getValue signal  */
SIGNALLIST  ODLSensInp=getOls, setOlsPeriod, setOlsEvent, setOlsPoll; /*    */
/* Signals at object output oo */
SIGNAL  newOls(Lux) ; /*    */
SIGNAL newODLMal;
SIGNALLIST  ODLSensOutp=ident,  newOls, newODLMal ; /*    */
```

MotDet

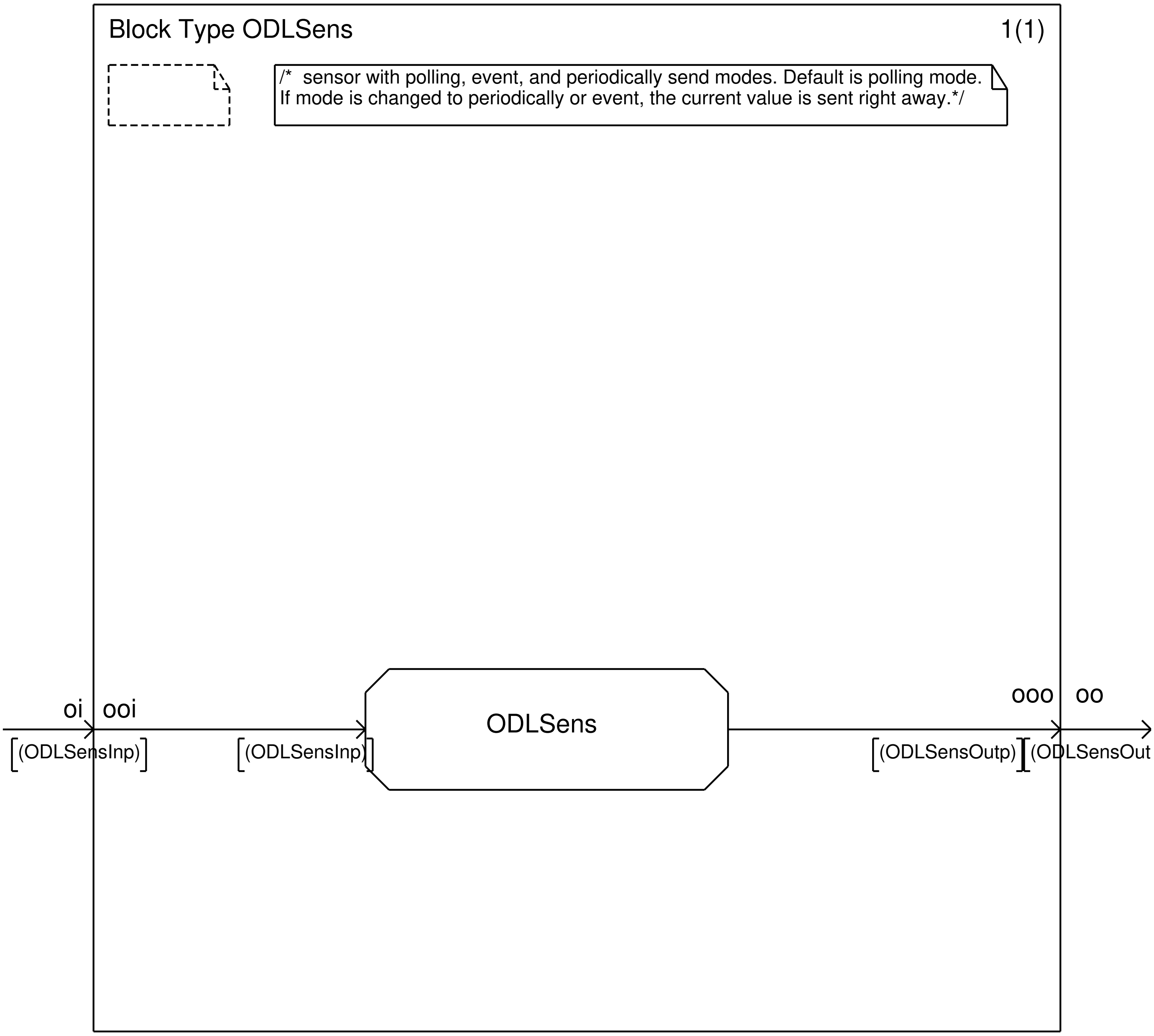
```
/* Signals at object input oi */
SIGNAL getMotDet ;  /*  Get value, independent of mode  */
SIGNAL setMDPeriod(Duration) ; /* Send value periodically with period p  */
SIGNAL setMDEvent ; /* Send value if changed  */
SIGNAL setMDPoll ; /* Wait for getValue signal  */
SIGNALLIST  MotDetInp=getMotDet, setMDPeriod, setMDEvent,  setMDPoll ; /*    */
/* Signals at object output oo */
SIGNAL  newMotDet(Binary) ; /*    */
SIGNAL newMotDetMal (ErrorType);
SIGNALLIST  MotDetOutp=ident, newMotDet, newMotDetMal; /*    */
```

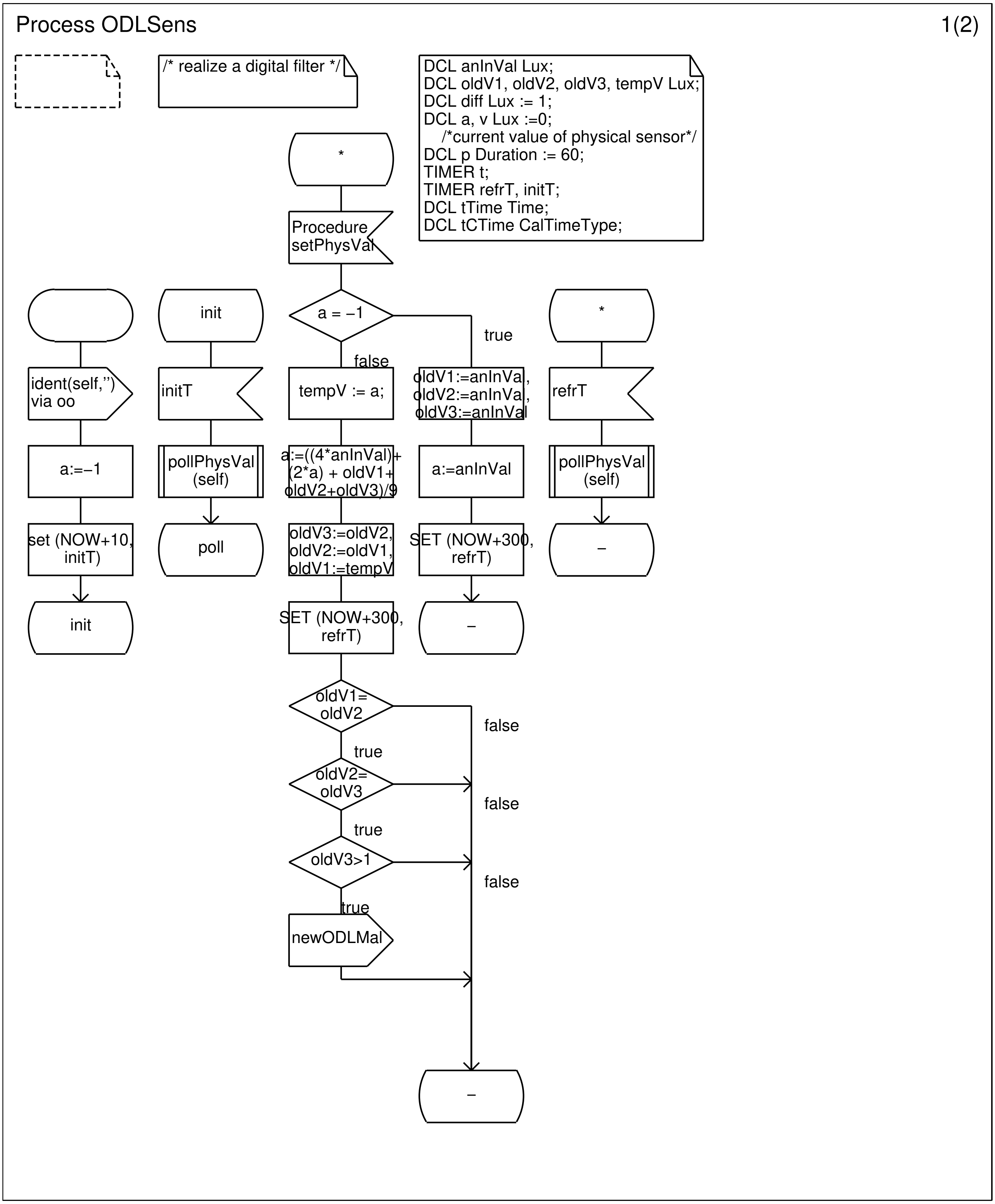
Switch

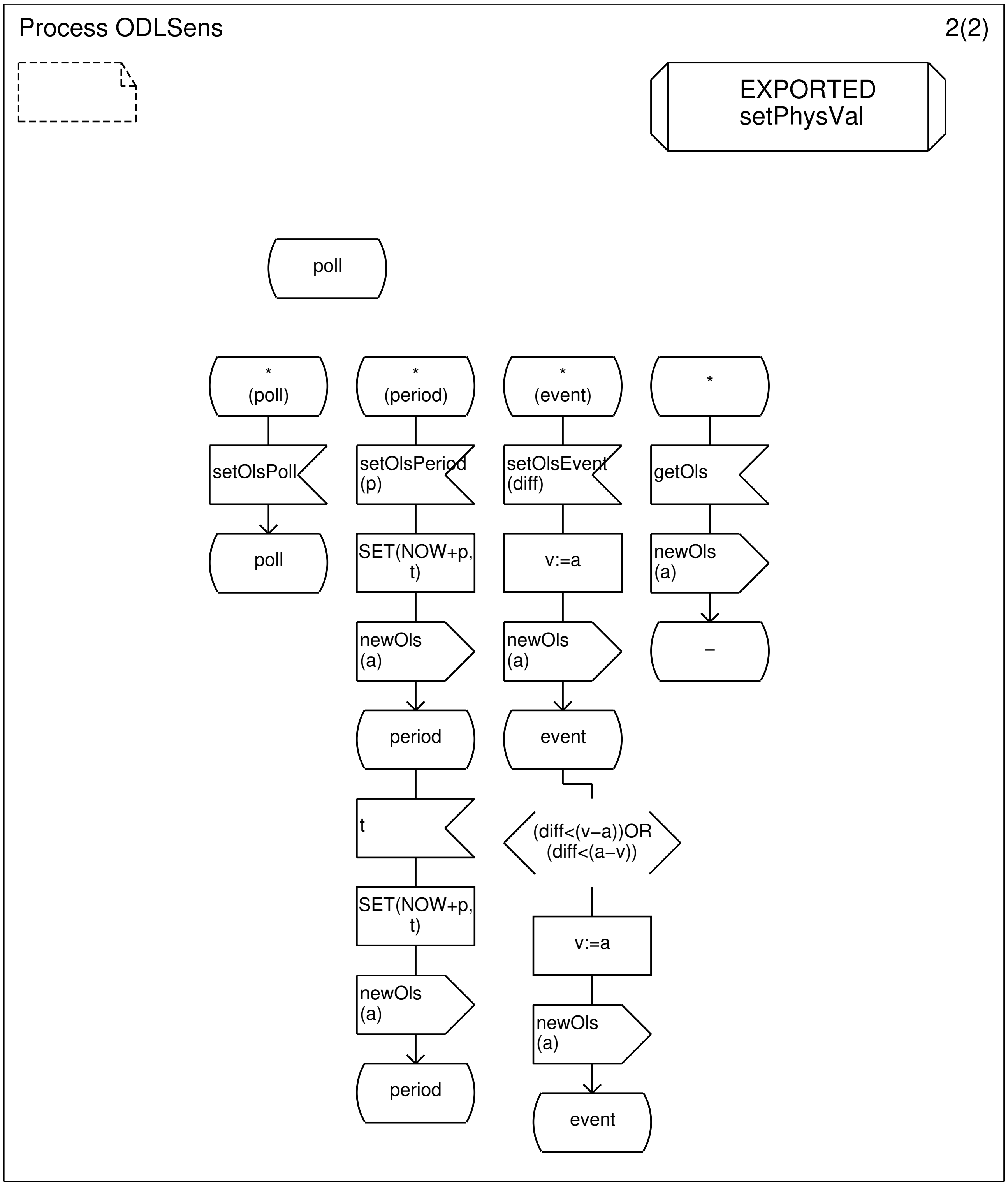
```
/* Signals at object input oi */
SIGNAL getSwitch ;  /*  Get value, independent of mode  */
SIGNAL setSwPeriod(Duration) ; /* Send value periodically with period p  */
SIGNAL setSwEvent ; /* Send value if changed  */
SIGNAL setSwPoll ; /* Wait for getValue signal  */
SIGNALLIST  SwitchInp=getSwitch, setSwPeriod, setSwEvent,  setSwPoll ; /*    */
/* Signals at object output oo */
SIGNAL  newSwitch(Binary) ; /*    */
SIGNAL newSwitchMal (ErrorType);
SIGNALLIST  SwitchOutp=ident, newSwitch, newSwitchMal; /*    */
```

SunDet

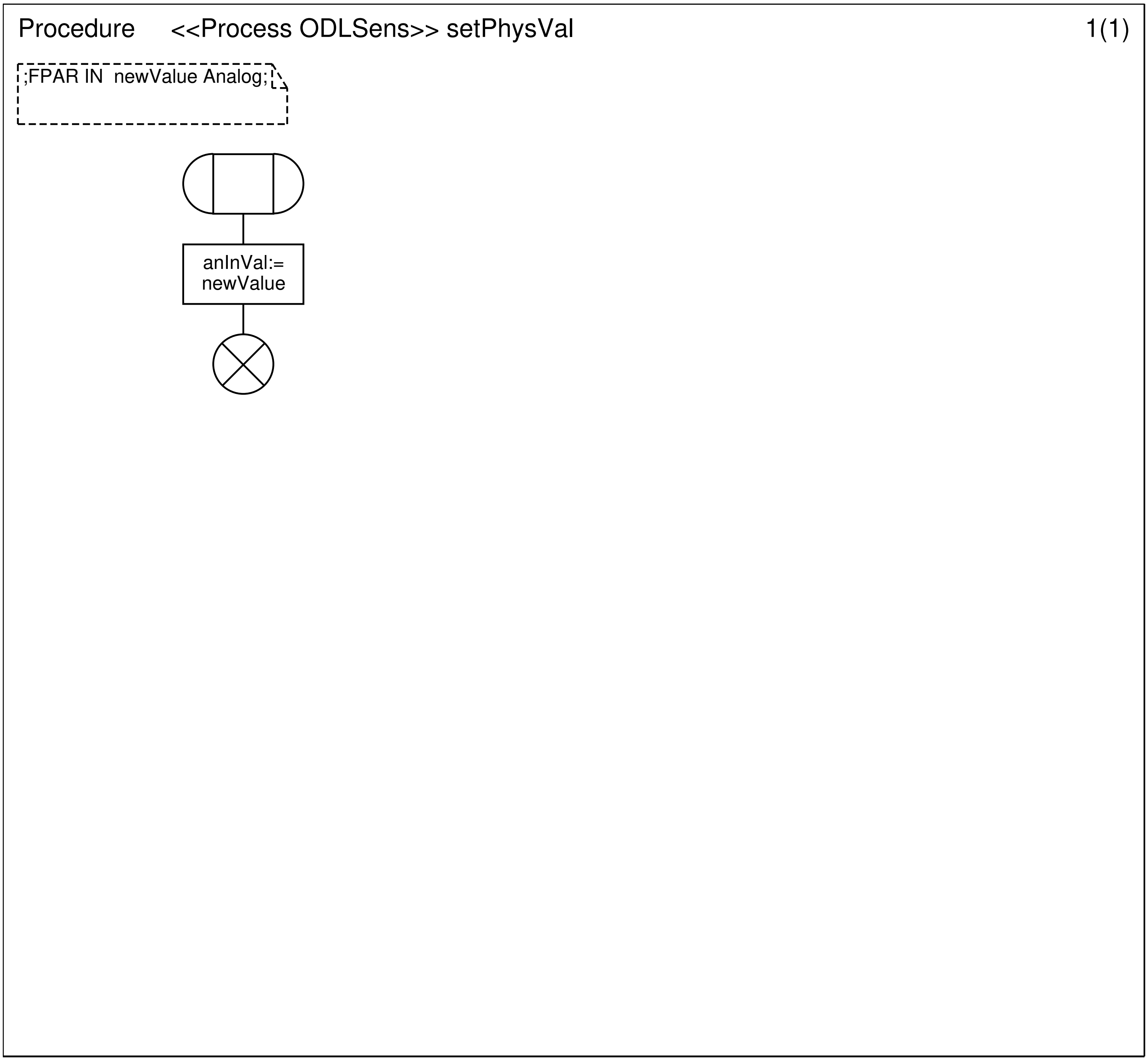
```
SIGNAL getSun (RoomIdType);
SIGNAL newSunDetMal (ErrorType);
SIGNAL newSun (RoomIdType, Lux);
SIGNALLIST SunDetInp = getSun;
SIGNALLIST SunDetOutp = ident, newSun, newSunDetMal;
```

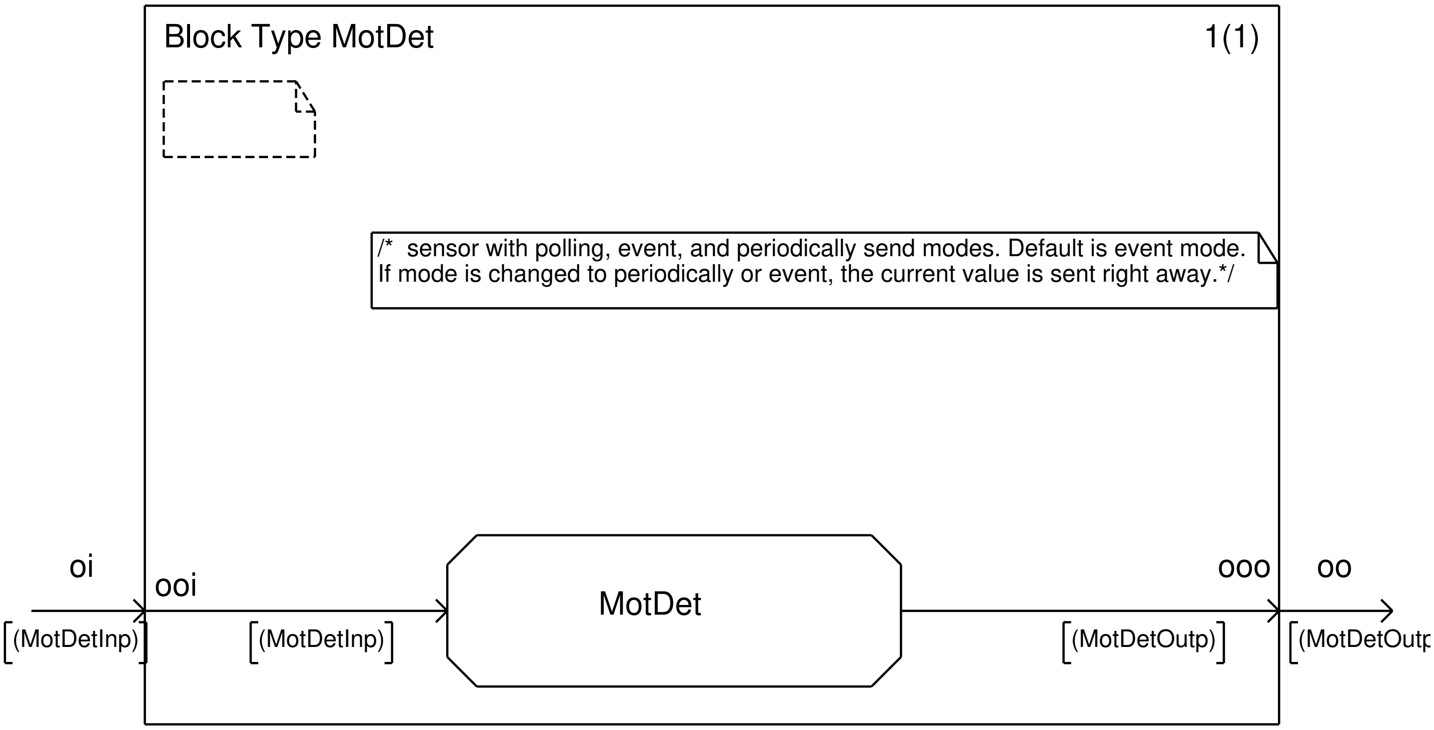

Block Type ODLSens
1(1)
/* sensor with polling, event, and periodically send modes. Default is polling mode.
If mode is changed to periodically or event, the current value is sent right away.*/
oi
ooi
(ODLSensInp)
(ODLSensInp)
ODLSens
ooo
oo
(ODLSensOutp)
(ODLSensOut

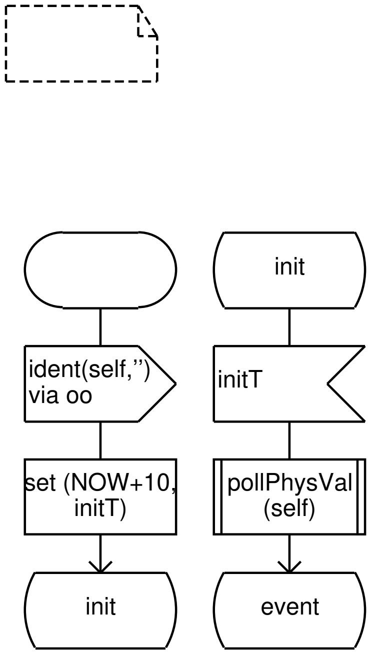

Process ODLSens
1(2)
/* realize a digital filter */
DCL anInVal Lux;
DCL oldV1, oldV2, oldV3, tempV Lux;
DCL diff Lux := 1;
DCL a, v Lux :=0;
/*current value of physical sensor*/
DCL p Duration := 60;
TIMER t;
TIMER refrT, initT;
DCL tTime Time;
DCL tCTime CalTimeType;
*
Procedure setPhysVal
a = −1
true
false
tempV := a;
oldV1:=anInVal, oldV2:=anInVal, oldV3:=anInVal
a:=((4*anInVal)+ (2*a) + oldV1+ oldV2+oldV3)/9
a:=anInVal
oldV3:=oldV2, oldV2:=oldV1, oldV1:=tempV
SET (NOW+300, refrT)
SET (NOW+300, refrT)
–
oldV1= oldV2
false
true
oldV2= oldV3
false
true
oldV3>1
false
true
newODLMal
–
ident(self,'') via oo
a:=−1
set (NOW+10, initT)
init
init
initT
pollPhysVal (self)
poll
*
refrT
pollPhysVal (self)
–

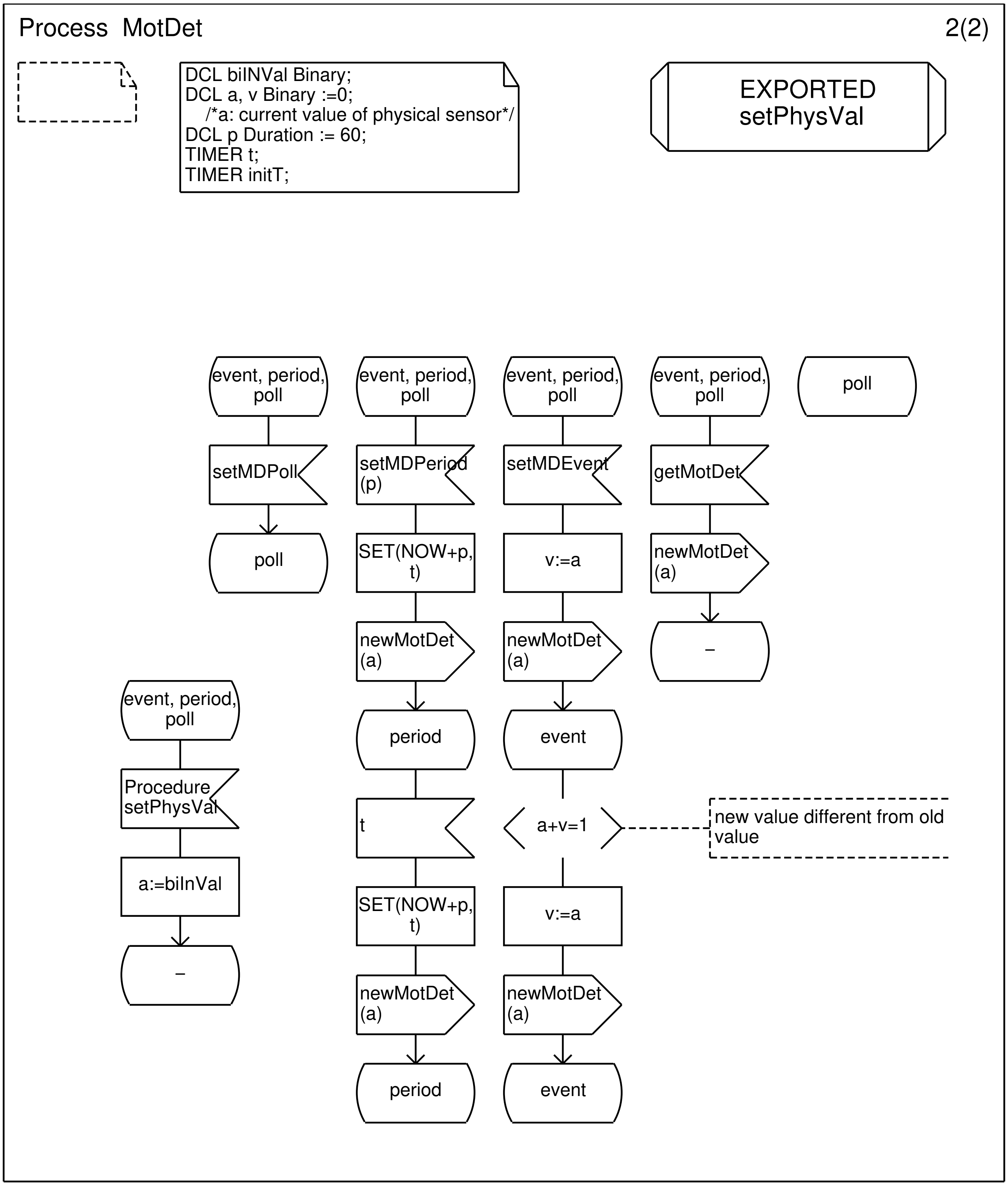


Process ODLSens
2(2)
EXPORTED
setPhysVal
poll
*
(poll)
setOlsPoll
poll
*
(period)
setOlsPeriod
(p)
SET(NOW+p,
t)
newOls
(a)
period
t
SET(NOW+p,
t)
newOls
(a)
period
*
(event)
setOlsEvent
(diff)
v:=a
newOls
(a)
event
(diff<(v–a))OR
(diff<(a–v))
v:=a
newOls
(a)
event
*
getOls
newOls
(a)
–

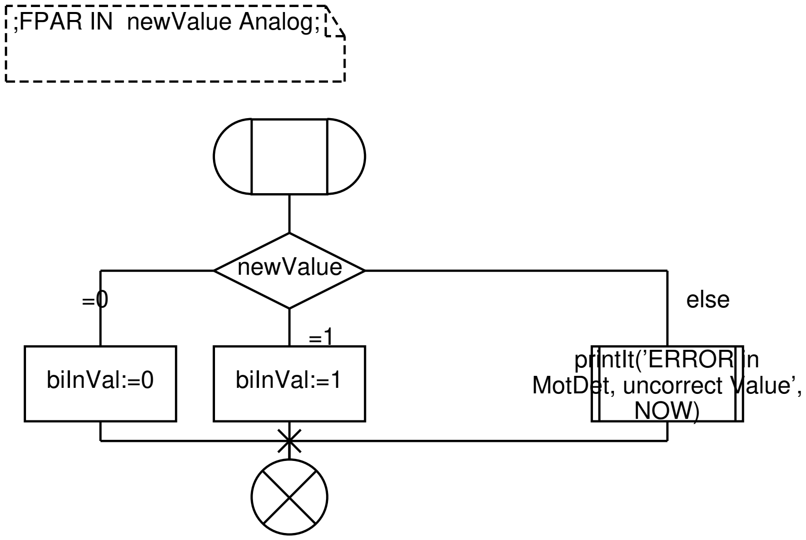

Procedure <<Process ODLSens>> setPhysVal
1(1)
;FPAR IN newValue Analog;
anInVal:=
newValue

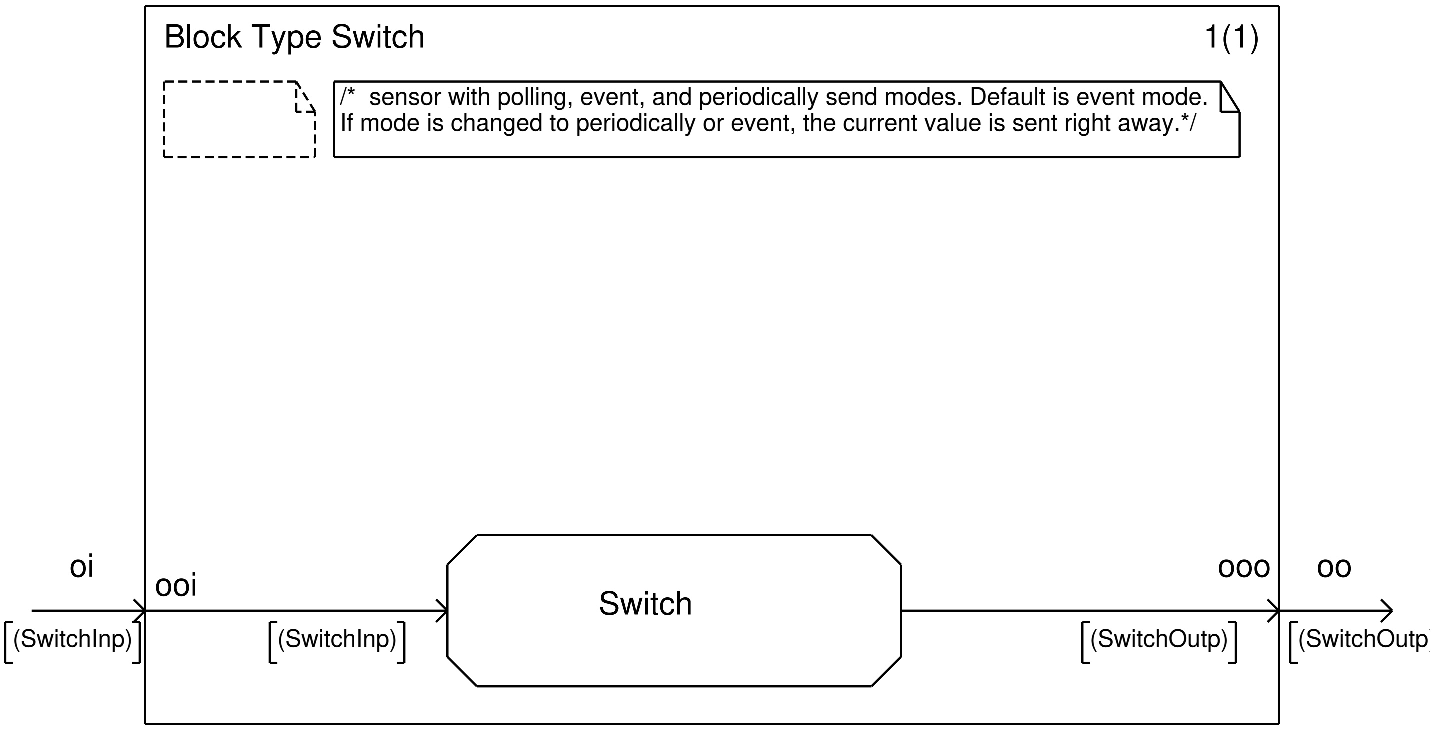

Block Type MotDet
1(1)
/* sensor with polling, event, and periodically send modes. Default is event mode.
If mode is changed to periodically or event, the current value is sent right away.*/
oi
ooi
(MotDetInp)
(MotDetInp)
MotDet
ooo
oo
(MotDetOutp)
(MotDetOutp

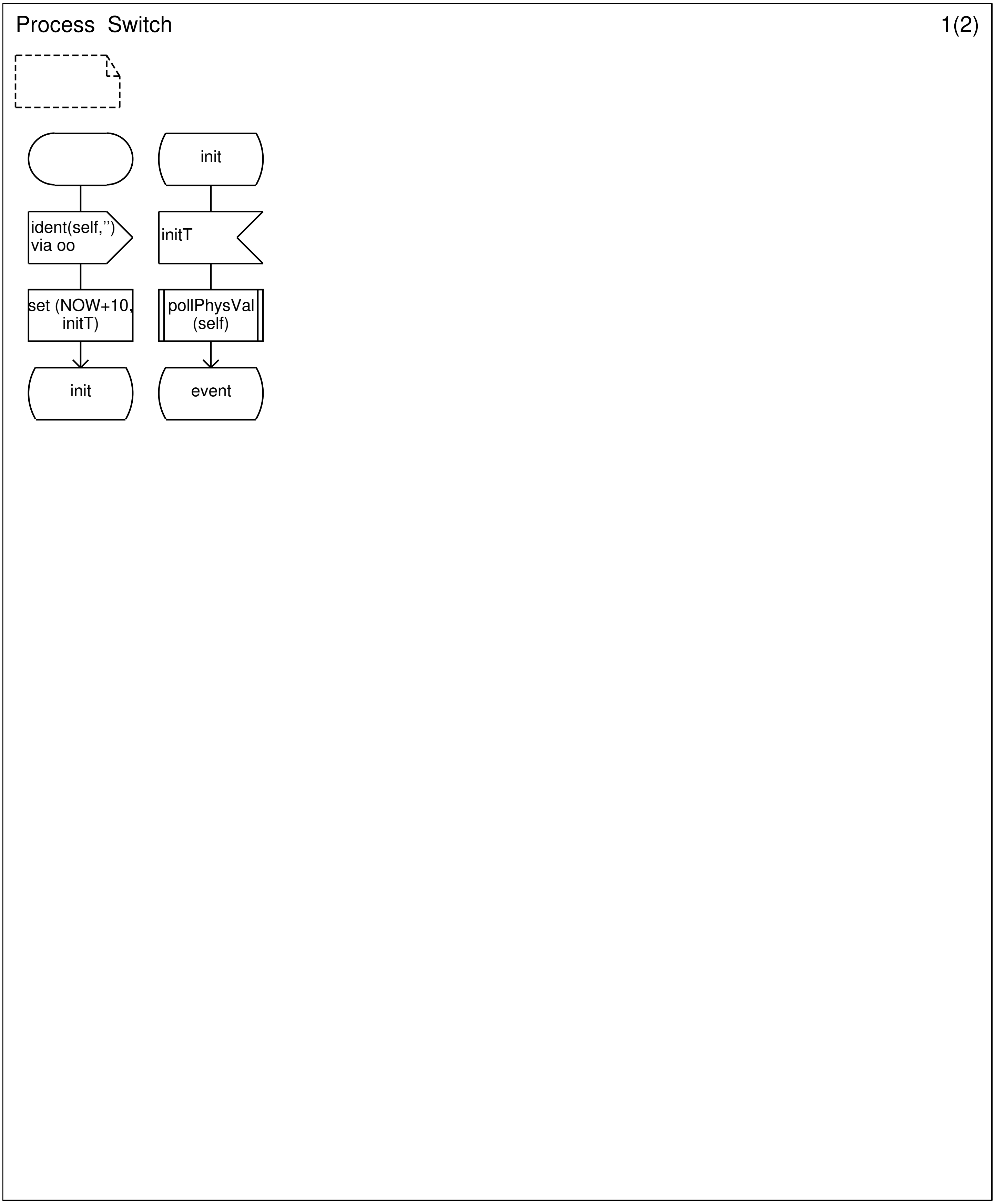

ident(self,'')
via oo
set (NOW+10,
initT)
init
init
initT
pollPhysVal
(self)
event

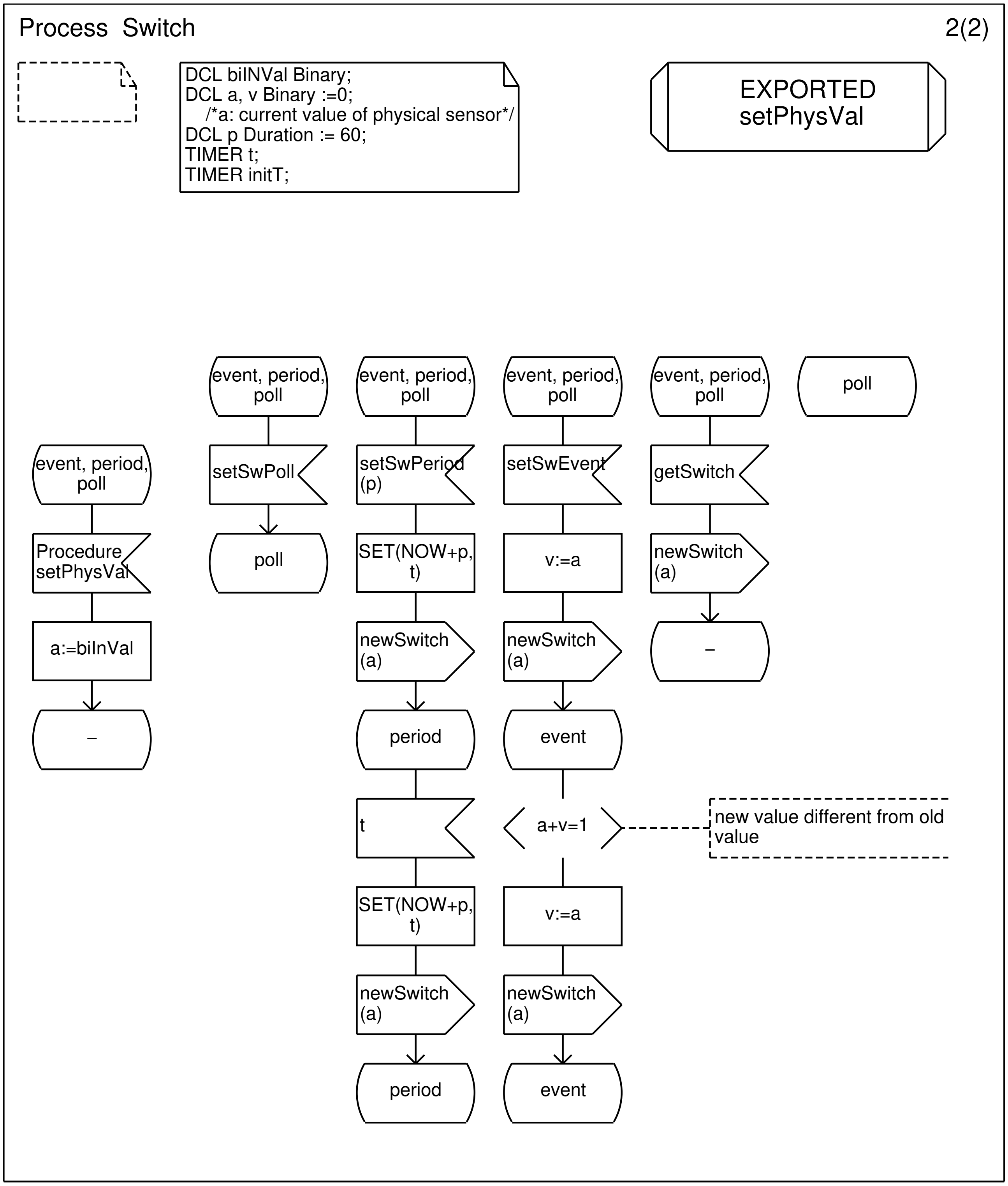

Process MotDet
2(2)
DCL biINVal Binary;
DCL a, v Binary :=0;
/*a: current value of physical sensor*/
DCL p Duration := 60;
TIMER t;
TIMER initT;
EXPORTED setPhysVal
event, period, poll
setMDPoll
poll
event, period, poll
setMDPeriod (p)
SET(NOW+p, t)
newMotDet (a)
period
t
SET(NOW+p, t)
newMotDet (a)
period
event, period, poll
setMDEvent
v:=a
newMotDet (a)
event
a+v=1
new value different from old value
v:=a
newMotDet (a)
event
event, period, poll
getMotDet
newMotDet (a)
–
poll
event, period, poll
Procedure setPhysVal
a:=biInVal
–

Procedure <<Process MotDet>> setPhysVal 1(1)

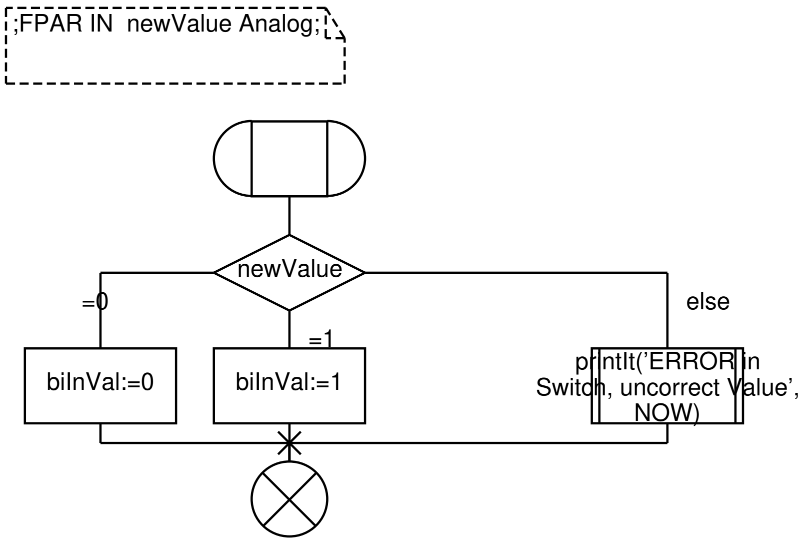



Block Type Switch
1(1)
/* sensor with polling, event, and periodically send modes. Default is event mode.
If mode is changed to periodically or event, the current value is sent right away.*/
oi
ooi
(SwitchInp)
(SwitchInp)
Switch
ooo
oo
(SwitchOutp)
(SwitchOutp


Process Switch
1(2)
init
ident(self,'')
via oo
initT
set (NOW+10,
initT)
pollPhysVal
(self)
init
event


Process Switch
2(2)
DCL biINVal Binary;
DCL a, v Binary :=0;
/*a: current value of physical sensor*/
DCL p Duration := 60;
TIMER t;
TIMER initT;
EXPORTED setPhysVal
event, period, poll
setSwPoll
poll
event, period, poll
setSwPeriod (p)
SET(NOW+p, t)
newSwitch (a)
period
t
SET(NOW+p, t)
newSwitch (a)
period
event, period, poll
setSwEvent
v:=a
newSwitch (a)
event
a+v=1
v:=a
newSwitch (a)
event
new value different from old value
event, period, poll
getSwitch
newSwitch (a)
–
poll
event, period, poll
Procedure setPhysVal
a:=biInVal
–

Procedure <<Process Switch>> setPhysVal 1(1)

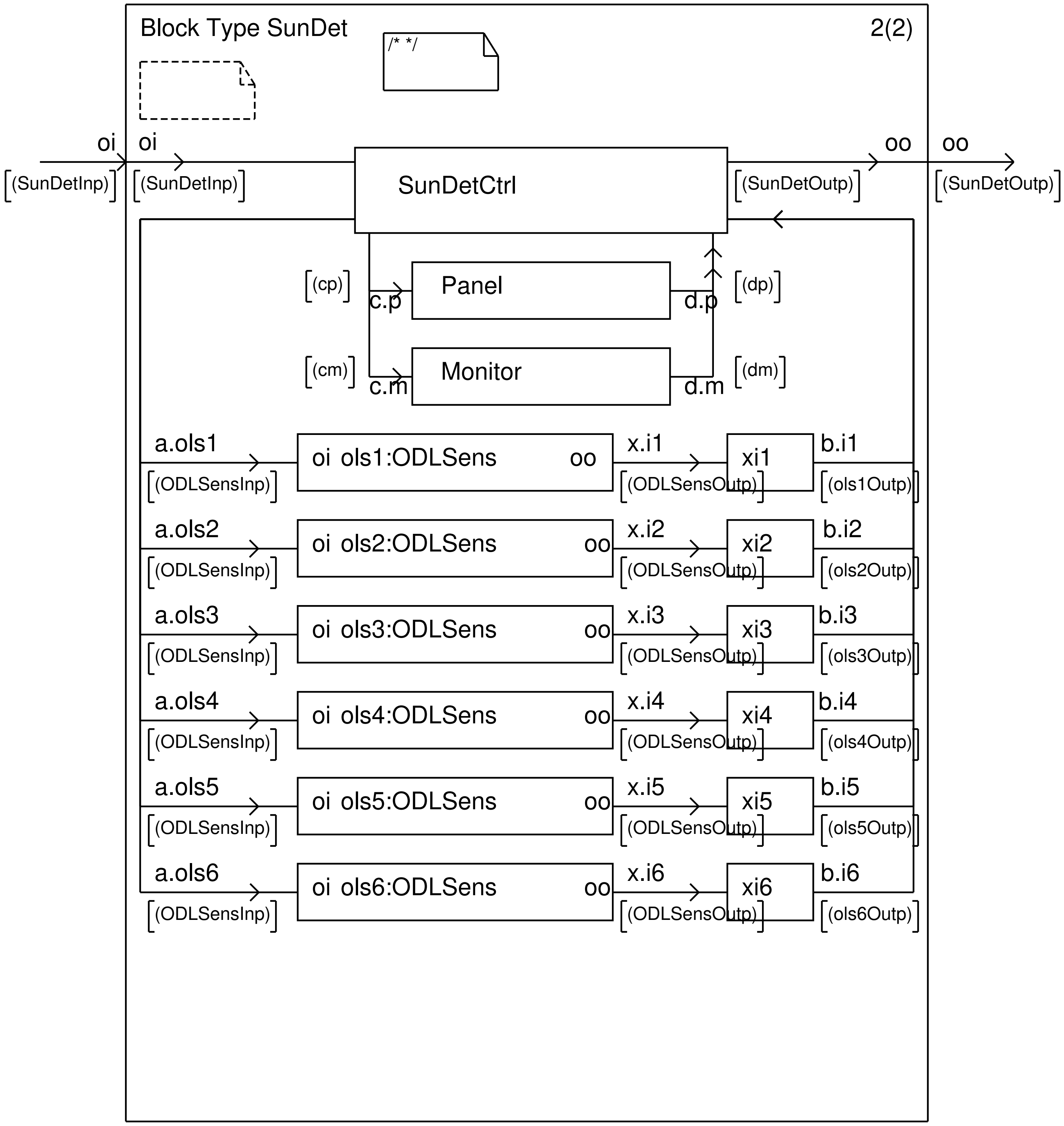


Block Type SunDet

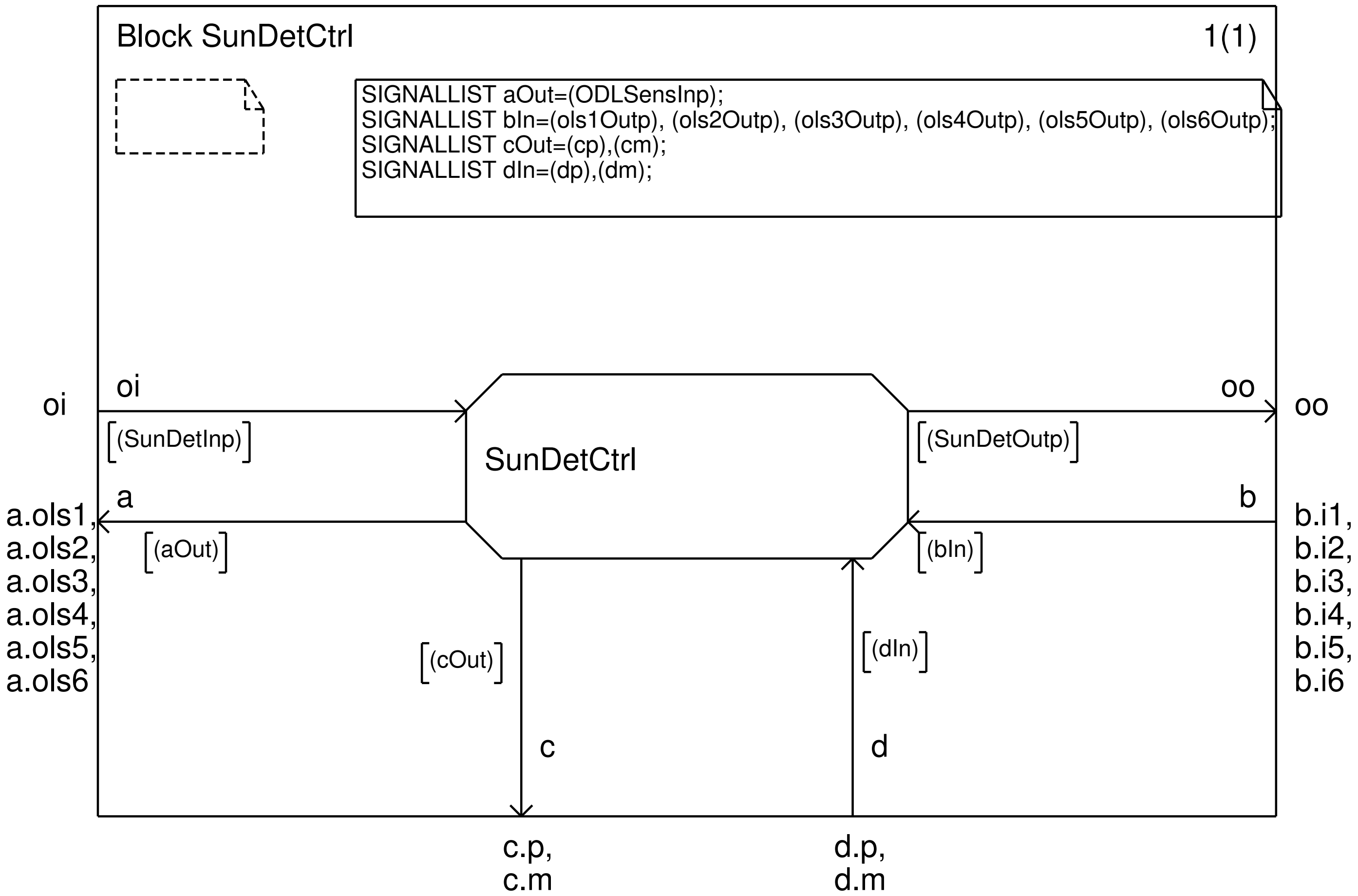

```
/*  */
```

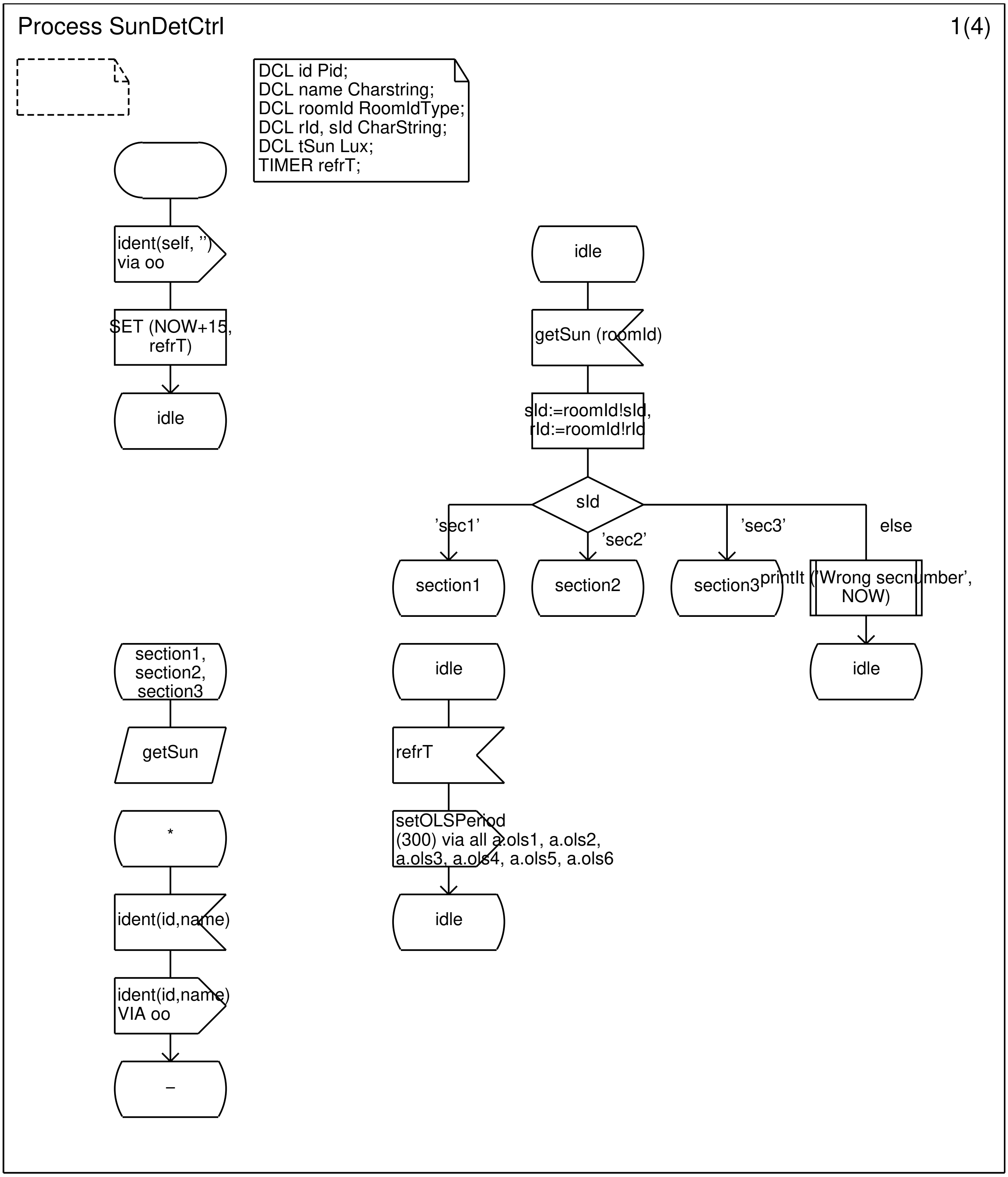
```
/* Signals to and from panel and monitor */
SIGNALLIST cp = dummy ; /*   */
SIGNALLIST cm = dummy ; /*   */
SIGNALLIST dp = dummy ; /*   */
SIGNALLIST dm = dummy ; /*   */
```

```
/* Signals from subobjects via b */
SIGNAL newOlsOls1 (Lux);
SIGNAL newODLMalOls1;
SIGNALLIST ols1Outp = ident, newOlsOls1, newODLMalOls1; /*   */
SIGNAL newODLMalOls2;
SIGNAL newOlsOls2 (Lux);
SIGNALLIST ols2Outp = ident, newOlsOls2, newODLMalOls2; /*   */
SIGNAL newODLMalOls3;
SIGNAL newOlsOls3 (Lux);
SIGNALLIST ols3Outp = ident, newOlsOls3, newODLMalOls3; /*   */
SIGNAL newODLMalOls4;
SIGNAL newOlsOls4 (Lux);
SIGNALLIST ols4Outp = ident, newOlsOls4, newODLMalOls4; /*   */
SIGNAL newODLMalOls5;
SIGNAL newOlsOls5 (Lux);
SIGNALLIST ols5Outp = ident, newOlsOls5, newODLMalOls5; /*   */
SIGNAL newODLMalOls6;
SIGNAL newOlsOls6 (Lux);
SIGNALLIST ols6Outp = ident, newOlsOls6, newODLMalOls6; /*   */
```

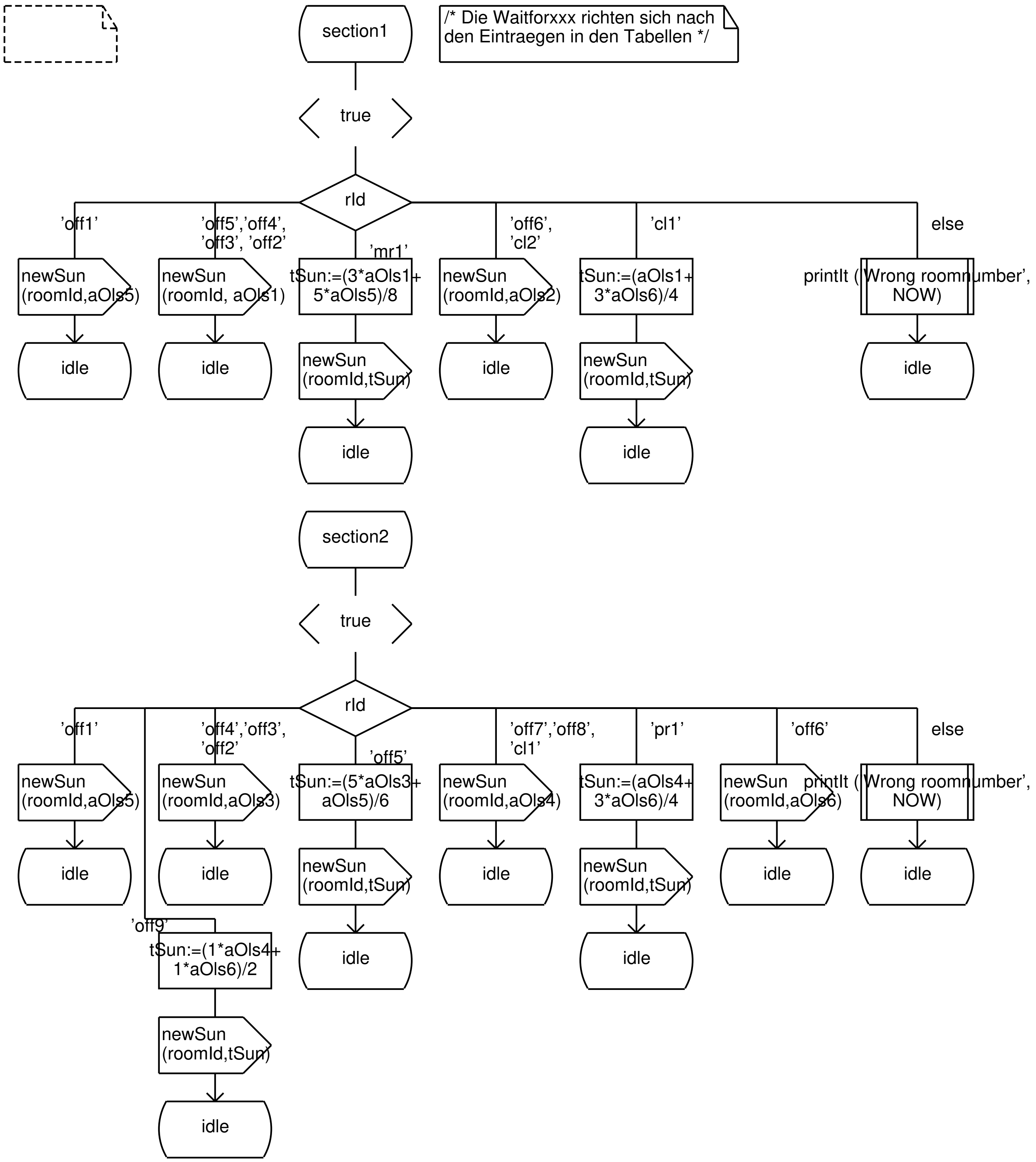


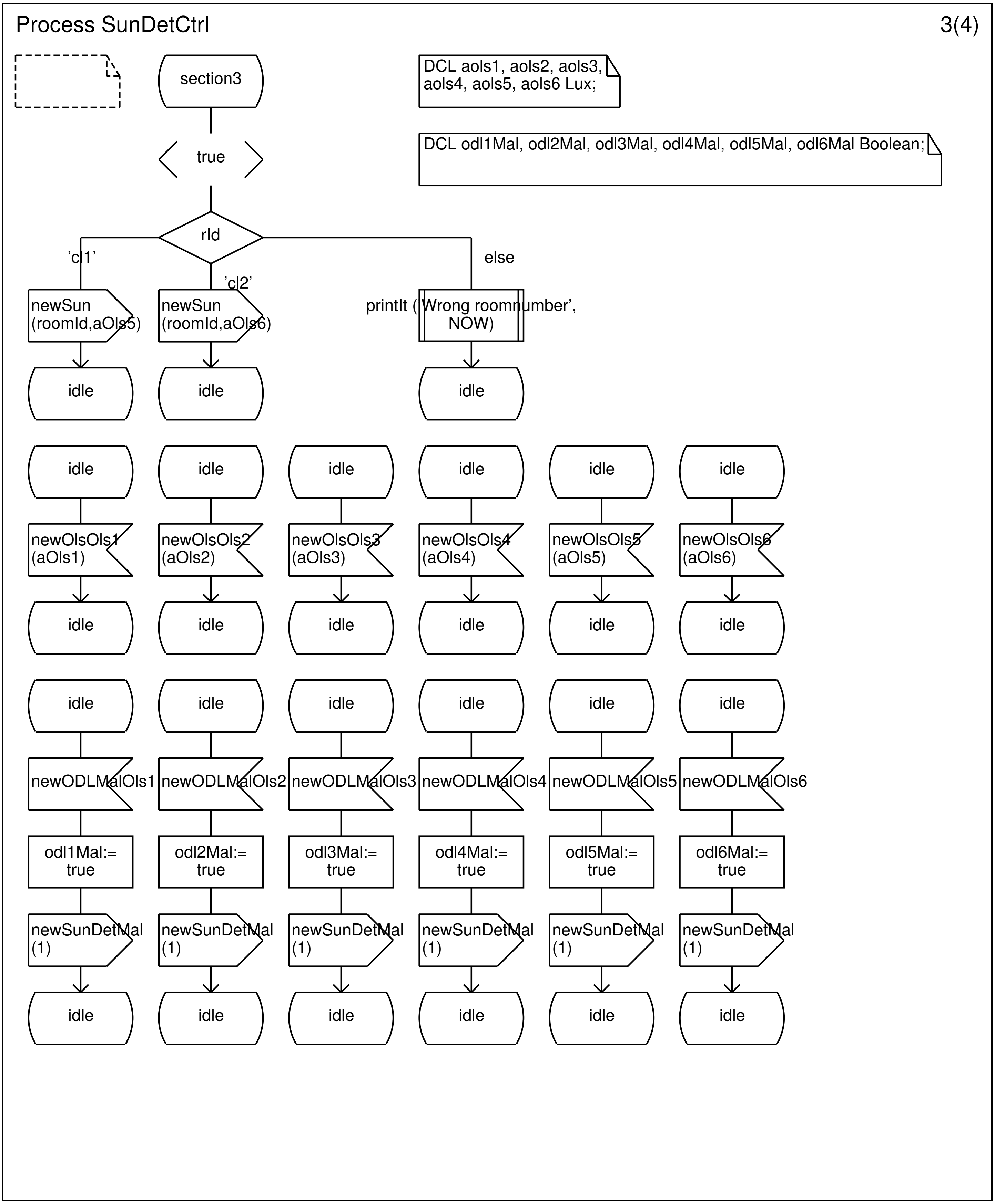
Block Type SunDet
2(2)
/* */
oi
oi
(SunDetInp)
(SunDetInp)
SunDetCtrl
oo
oo
(SunDetOutp)
(SunDetOutp)
(cp)
c.p
Panel
d.p
(dp)
(cm)
c.m
Monitor
d.m
(dm)
a.ols1
(ODLSensInp)
oi ols1:ODLSens oo
x.i1
(ODLSensOutp)
xi1
b.i1
(ols1Outp)
a.ols2
(ODLSensInp)
oi ols2:ODLSens oo
x.i2
(ODLSensOutp)
xi2
b.i2
(ols2Outp)
a.ols3
(ODLSensInp)
oi ols3:ODLSens oo
x.i3
(ODLSensOutp)
xi3
b.i3
(ols3Outp)
a.ols4
(ODLSensInp)
oi ols4:ODLSens oo
x.i4
(ODLSensOutp)
xi4
b.i4
(ols4Outp)
a.ols5
(ODLSensInp)
oi ols5:ODLSens oo
x.i5
(ODLSensOutp)
xi5
b.i5
(ols5Outp)
a.ols6
(ODLSensInp)
oi ols6:ODLSens oo
x.i6
(ODLSensOutp)
xi6
b.i6
(ols6Outp)


Block SunDetCtrl
1(1)
SIGNALLIST aOut=(ODLSensInp);
SIGNALLIST bIn=(ols1Outp), (ols2Outp), (ols3Outp), (ols4Outp), (ols5Outp), (ols6Outp);
SIGNALLIST cOut=(cp),(cm);
SIGNALLIST dIn=(dp),(dm);
oi
oi
(SunDetInp)
SunDetCtrl
oo
oo
(SunDetOutp)
a
a.ols1,
a.ols2,
a.ols3,
a.ols4,
a.ols5,
a.ols6
(aOut)
b
b.i1,
b.i2,
b.i3,
b.i4,
b.i5,
b.i6
(bIn)
(cOut)
c
c.p,
c.m
(dIn)
d
d.p,
d.m

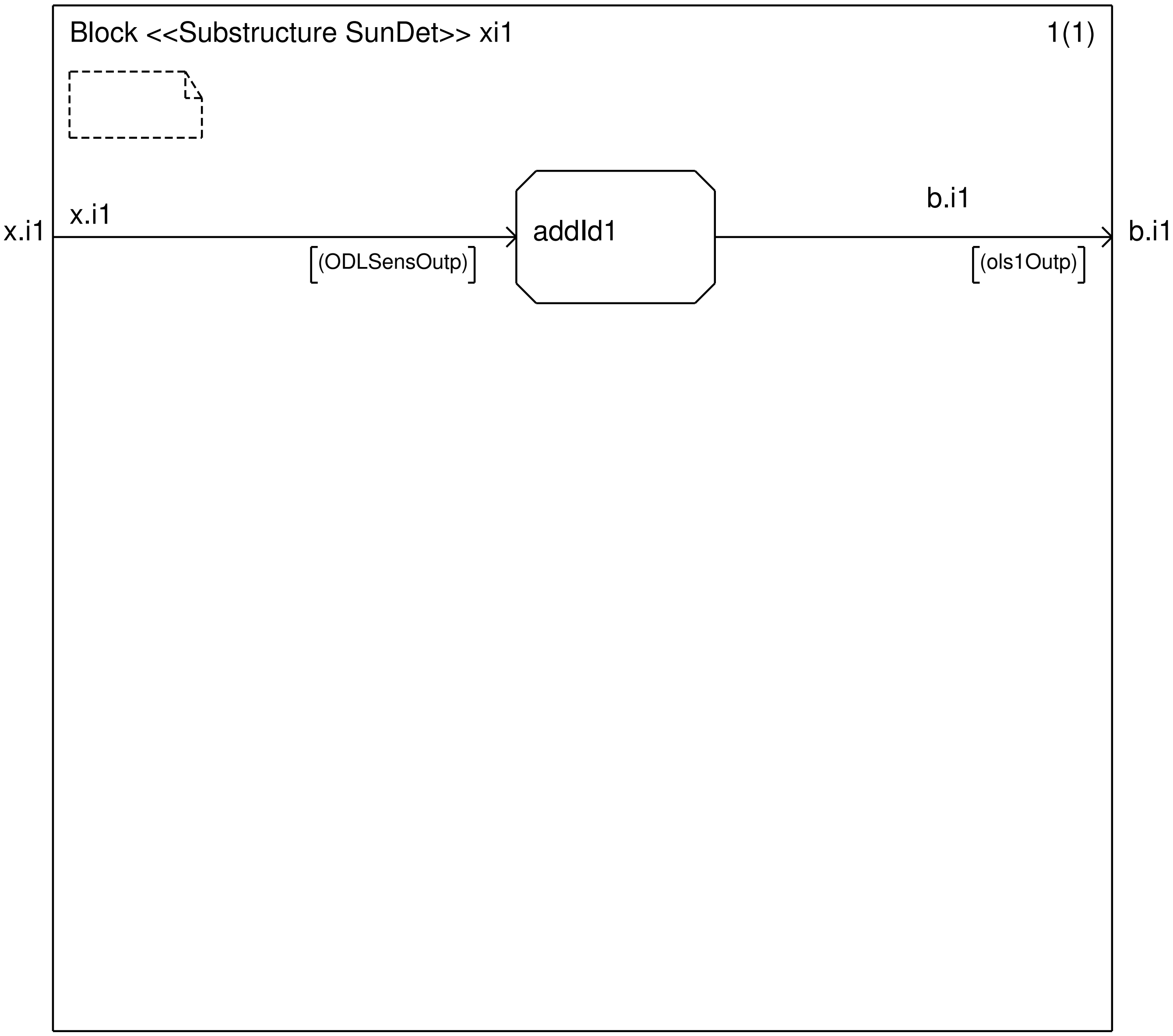

Process SunDetCtrl
1(4)
DCL id Pid;
DCL name Charstring;
DCL roomId RoomIdType;
DCL rId, sId CharString;
DCL tSun Lux;
TIMER refrT;
ident(self, '')
via oo
SET (NOW+15,
refrT)
idle
idle
getSun (roomId)
sId:=roomId!sId,
rId:=roomId!rId
sId
'sec1'
'sec2'
'sec3'
else
section1
section2
section3
printIt ('Wrong secnumber',
NOW)
idle
section1,
section2,
section3
getSun
idle
refrT
setOLSPeriod
(300) via all a.ols1, a.ols2,
a.ols3, a.ols4, a.ols5, a.ols6
idle
*
ident(id,name)
ident(id,name)
VIA oo
–

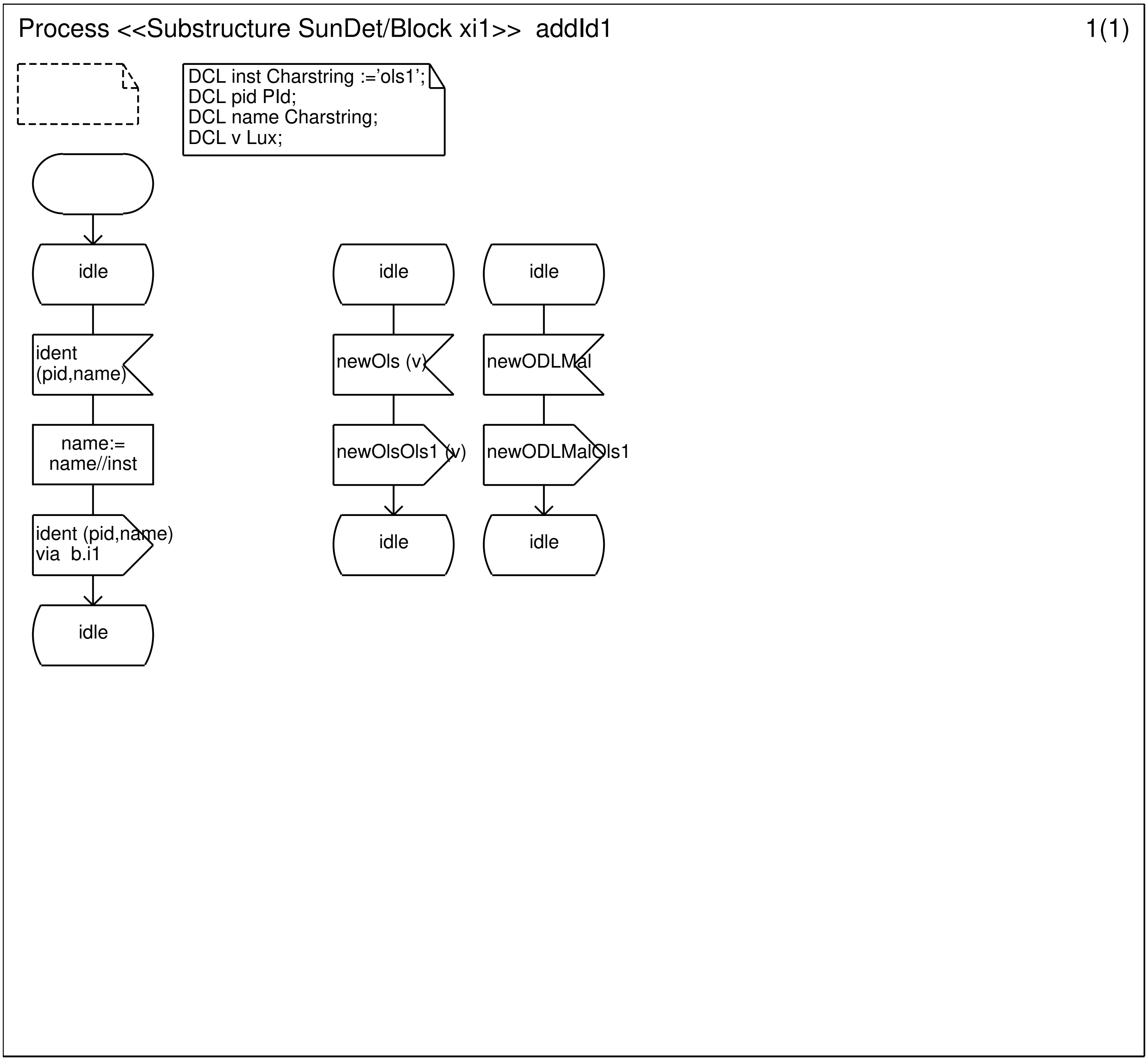
Process SunDetCtrl

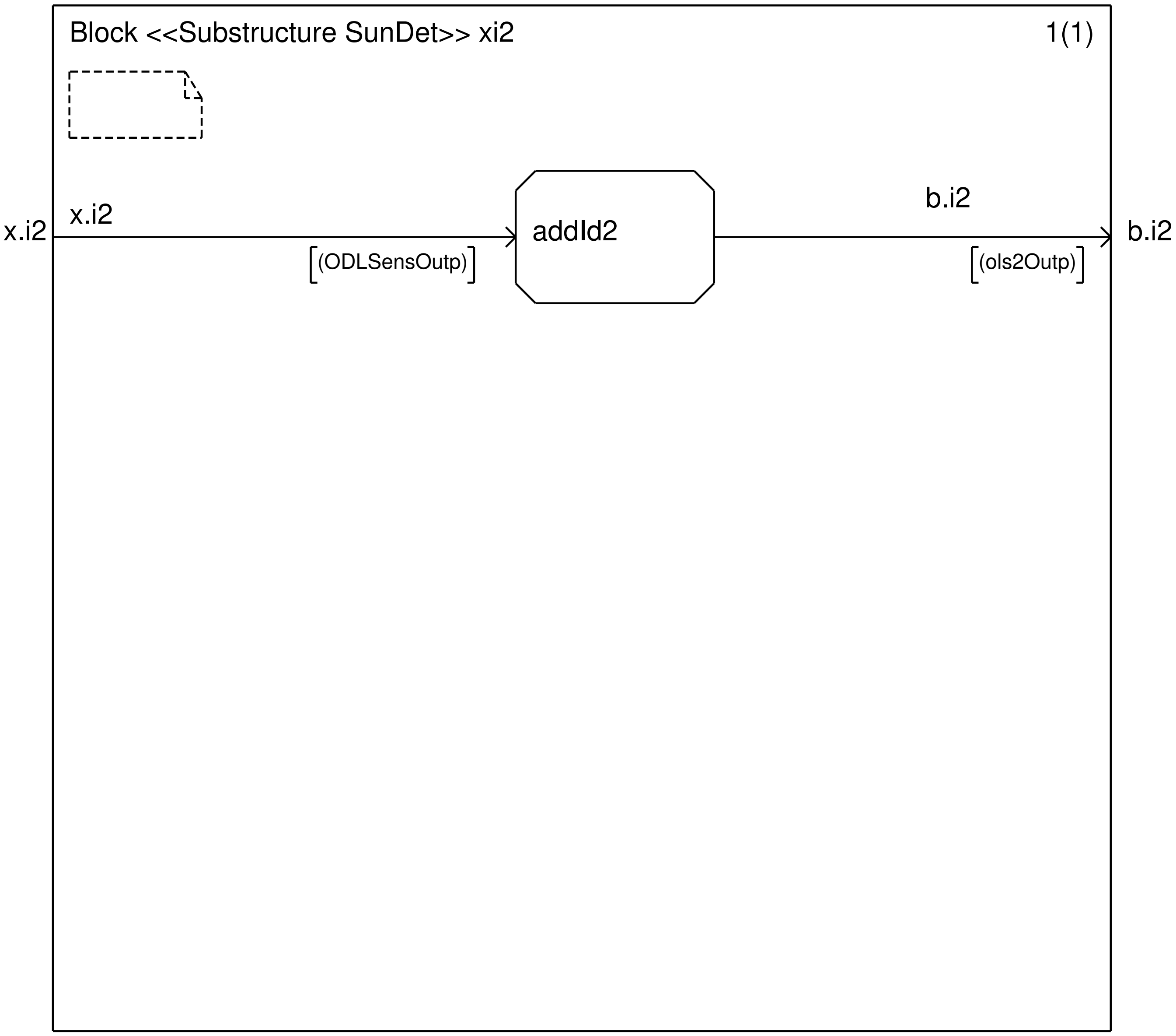


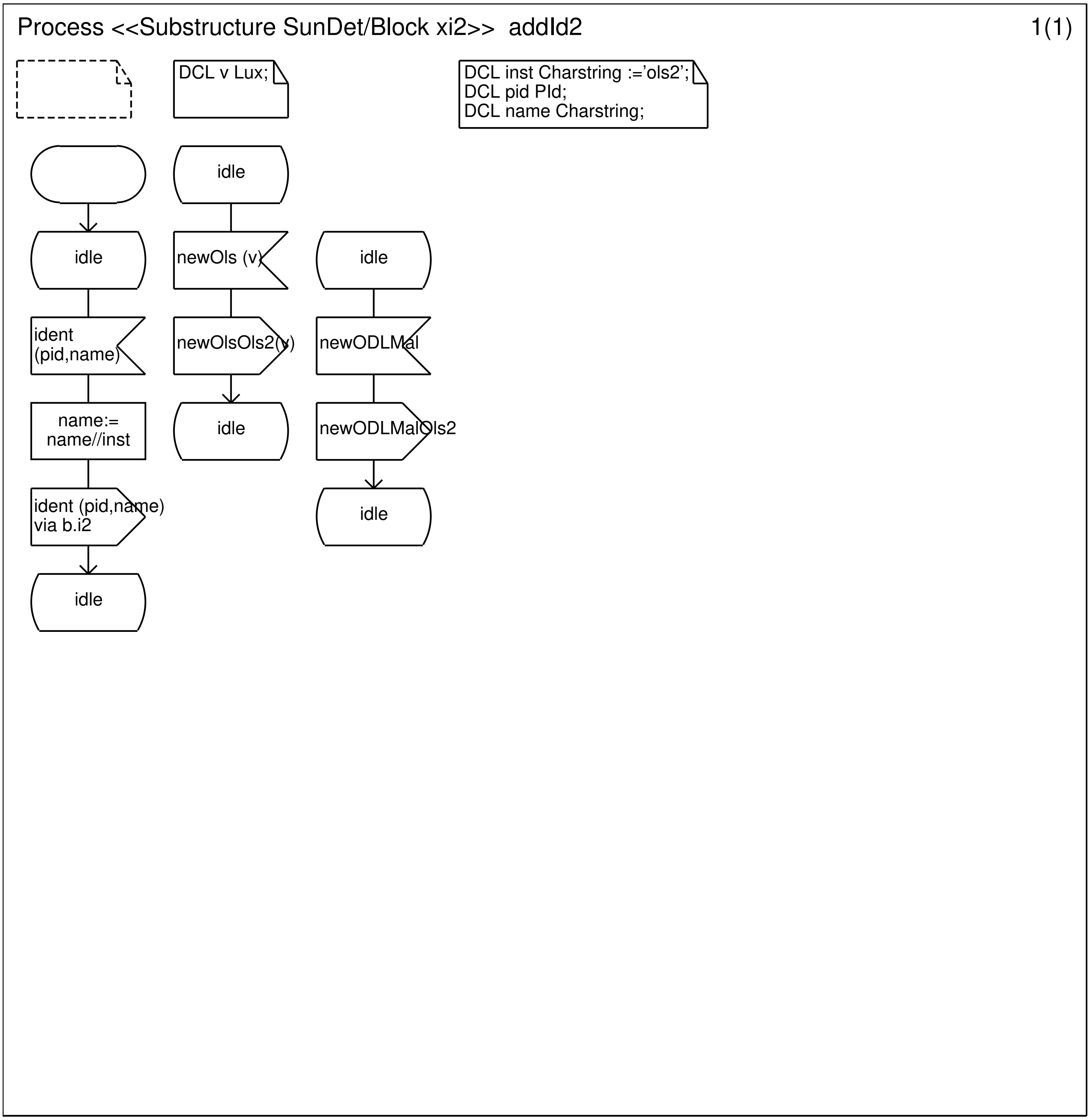

Process SunDetCtrl
3(4)
section3
DCL aols1, aols2, aols3, aols4, aols5, aols6 Lux;
DCL odl1Mal, odl2Mal, odl3Mal, odl4Mal, odl5Mal, odl6Mal Boolean;
true
rId
'cl1'
'cl2'
else
newSun (roomId,aOls5)
newSun (roomId,aOls6)
printIt ('Wrong roomnumber', NOW)
idle
newOlsOls1 (aOls1)
newOlsOls2 (aOls2)
newOlsOls3 (aOls3)
newOlsOls4 (aOls4)
newOlsOls5 (aOls5)
newOlsOls6 (aOls6)
newODLMalOls1
newODLMalOls2
newODLMalOls3
newODLMalOls4
newODLMalOls5
newODLMalOls6
odl1Mal:= true
odl2Mal:= true
odl3Mal:= true
odl4Mal:= true
odl5Mal:= true
odl6Mal:= true
newSunDetMal (1)

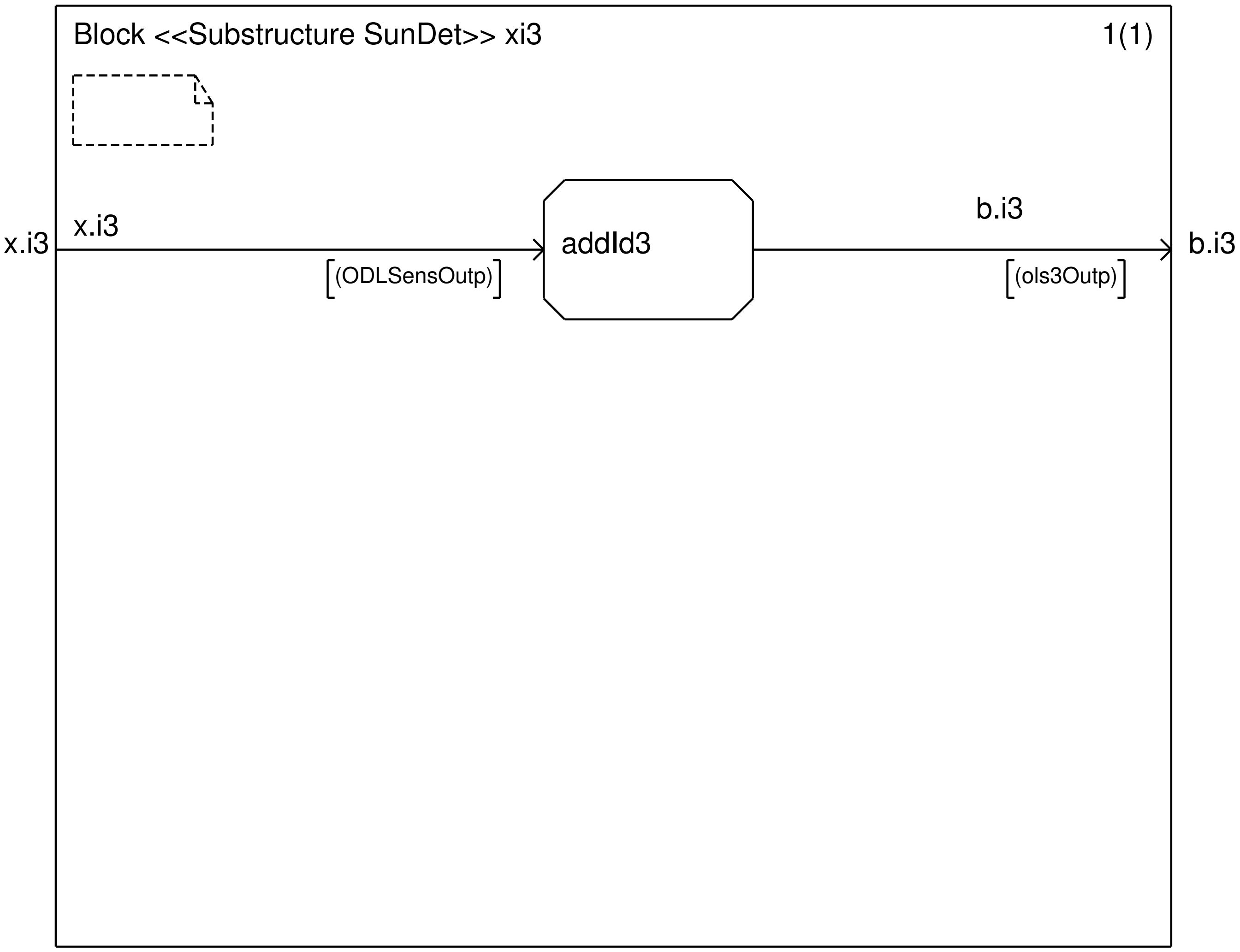
Process SunDetCtrl

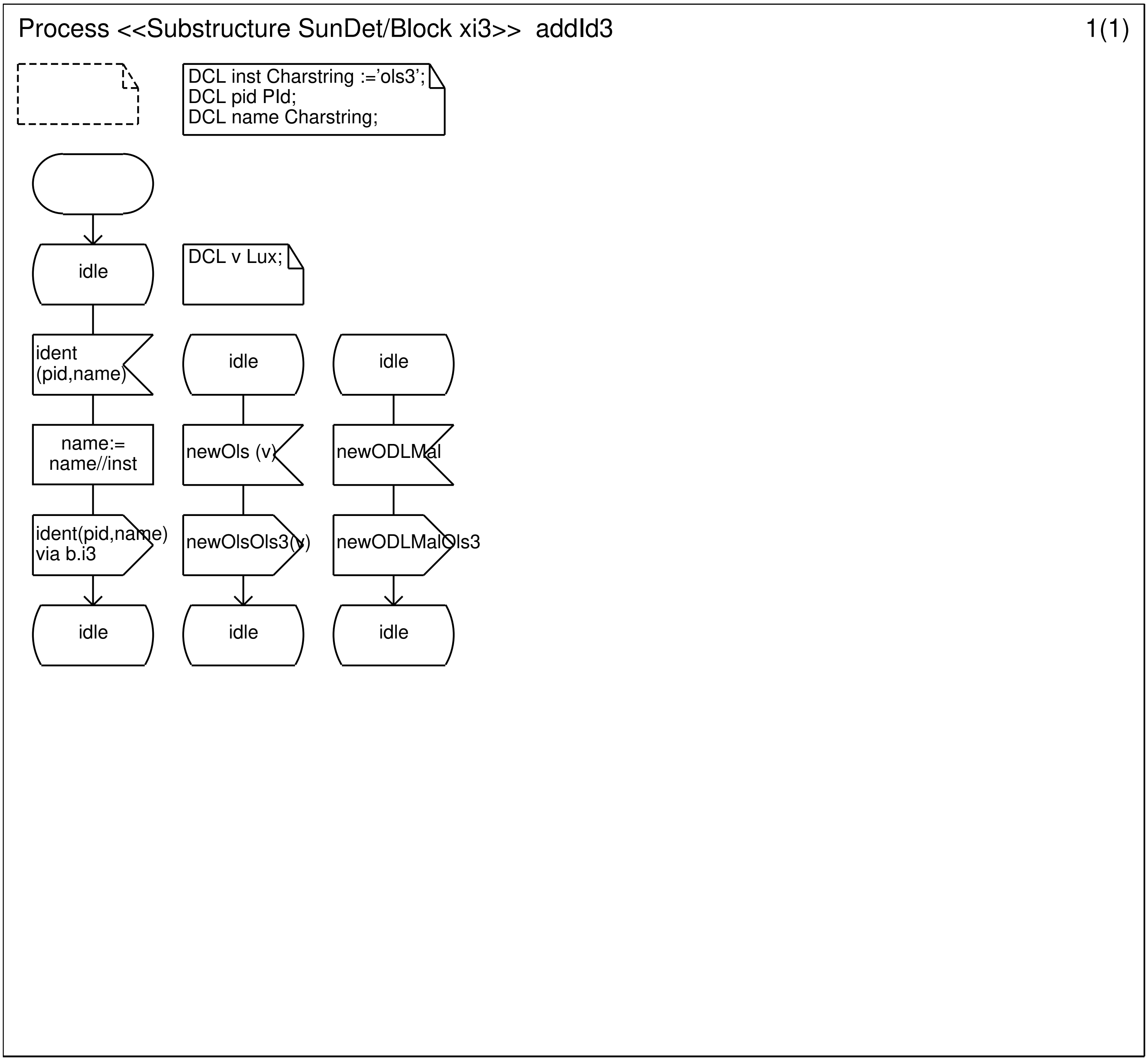

Block <<Substructure SunDet>> xi1
1(1)
x.i1
x.i1
(ODLSensOutp)
addId1
b.i1
b.i1
(ols1Outp)

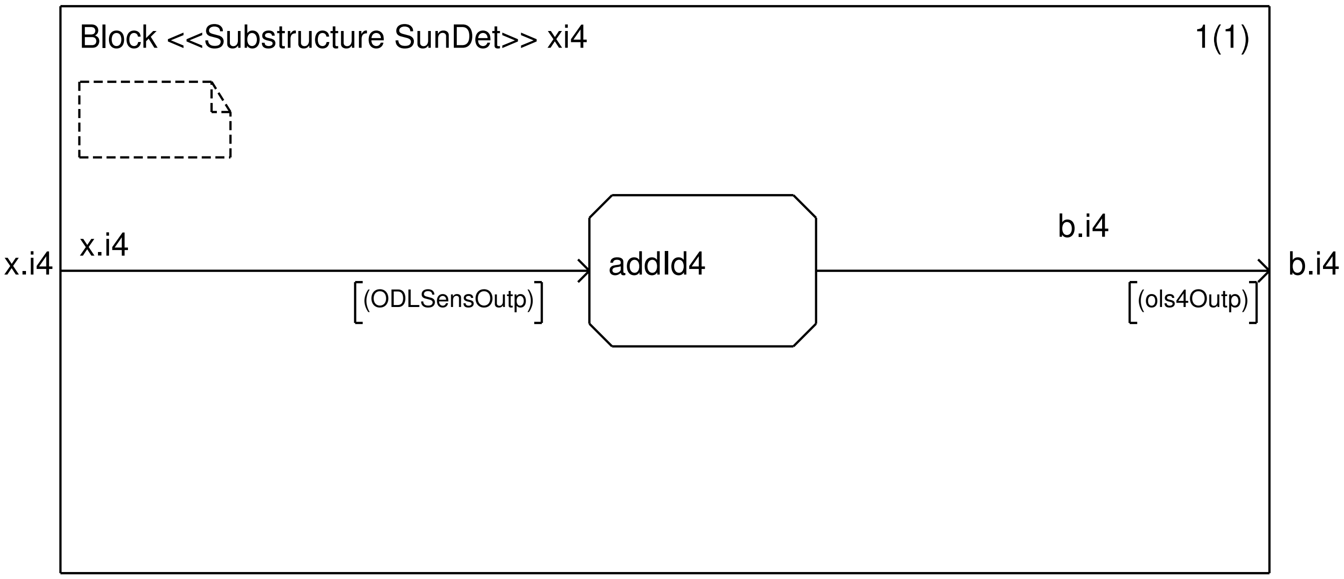

Process <<Substructure SunDet/Block xi1>> addId1
1(1)
DCL inst Charstring :='ols1';
DCL pid PId;
DCL name Charstring;
DCL v Lux;
idle
ident
(pid,name)
name:=
name//inst
ident (pid,name)
via b.i1
idle
idle
newOls (v)
newOlsOls1 (v)
idle
idle
newODLMal
newODLMalOls1
idle

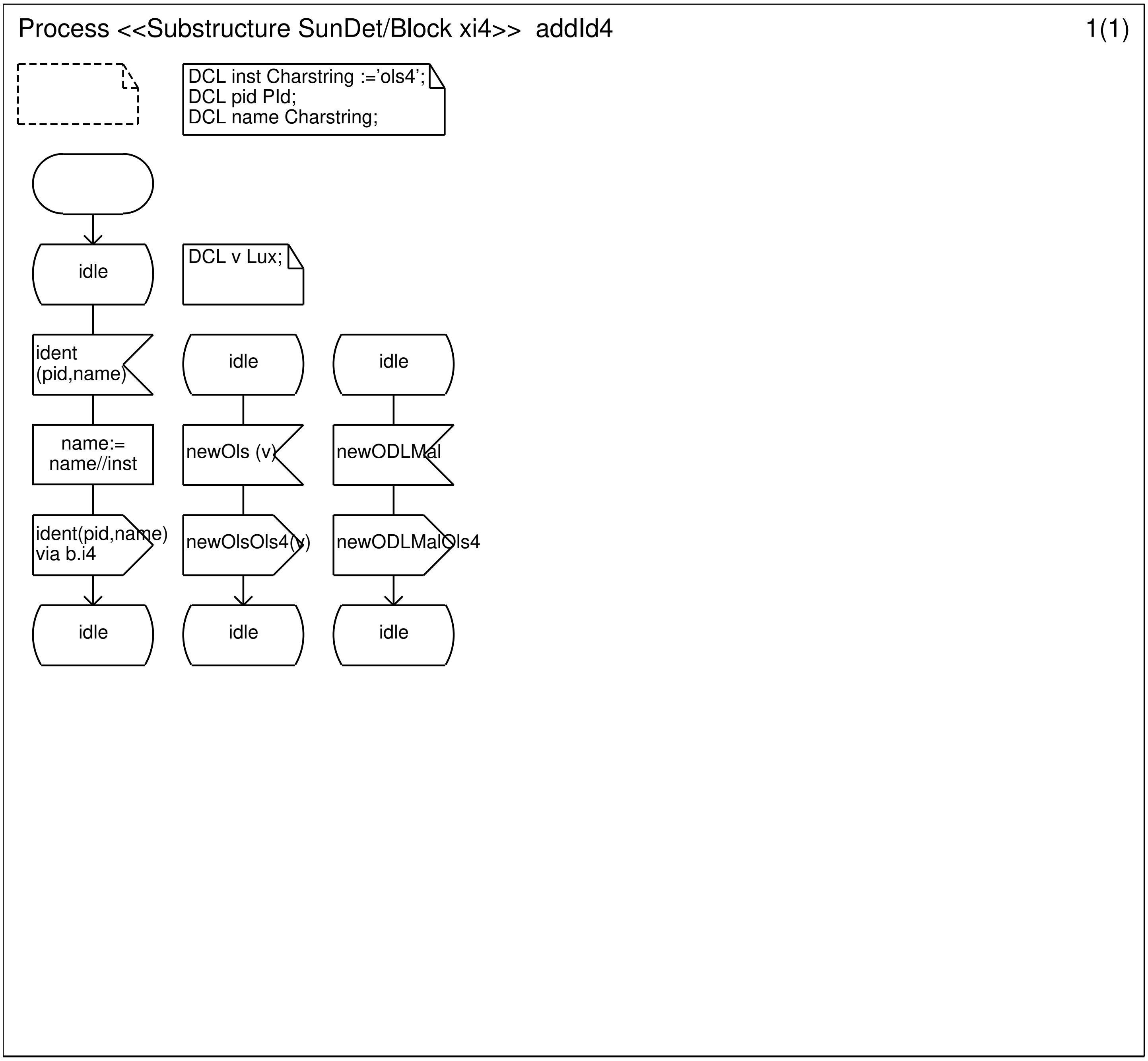

Block <<Substructure SunDet>> xi2
1(1)
x.i2
x.i2
(ODLSensOutp)
addId2
b.i2
b.i2
(ols2Outp)


Process <<Substructure SunDet/Block xi2>> addId2
1(1)
DCL v Lux;
DCL inst Charstring :='ols2';
DCL pid PId;
DCL name Charstring;
idle
idle
newOls (v)
idle
ident
(pid,name)
newOlsOls2(v)
newODLMal
name:=
name//inst
idle
newODLMalOls2
ident (pid,name)
via b.i2
idle
idle

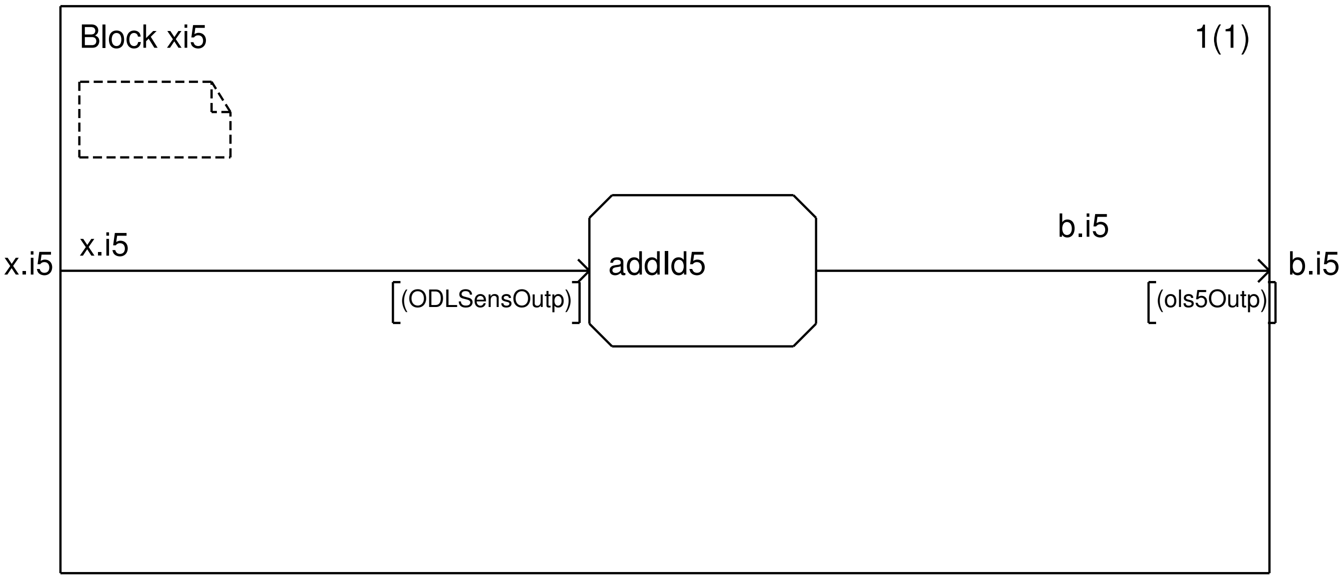

Block <<Substructure SunDet>> xi3
1(1)
x.i3
x.i3
(ODLSensOutp)
addId3
b.i3
b.i3
(ols3Outp)

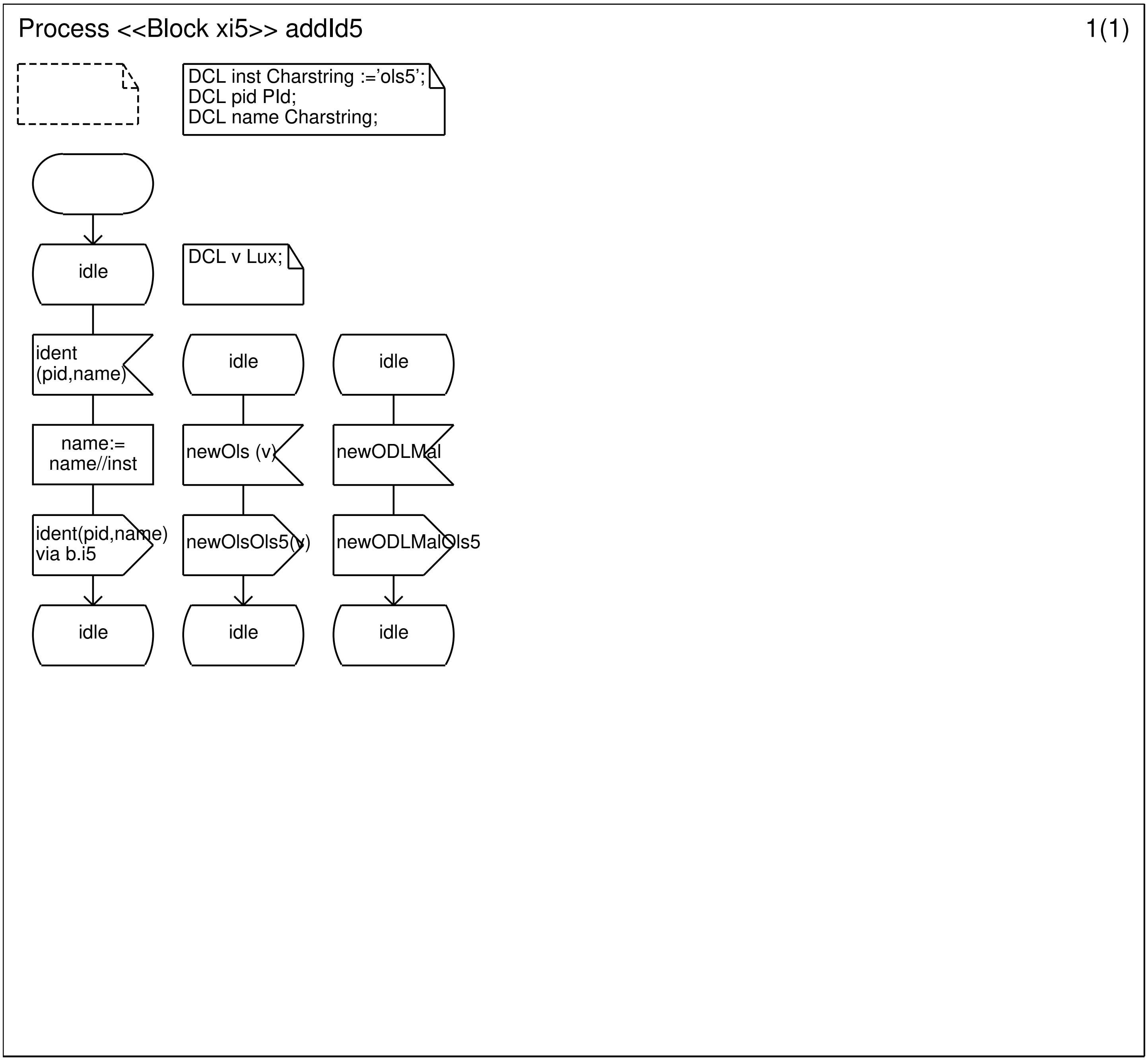

Process <<Substructure SunDet/Block xi3>> addId3
1(1)
DCL inst Charstring :='ols3';
DCL pid PId;
DCL name Charstring;
idle
DCL v Lux;
ident
(pid,name)
idle
idle
name:=
name//inst
newOls (v)
newODLMal
ident(pid,name)
via b.i3
newOlsOls3(v)
newODLMalOls3
idle
idle
idle

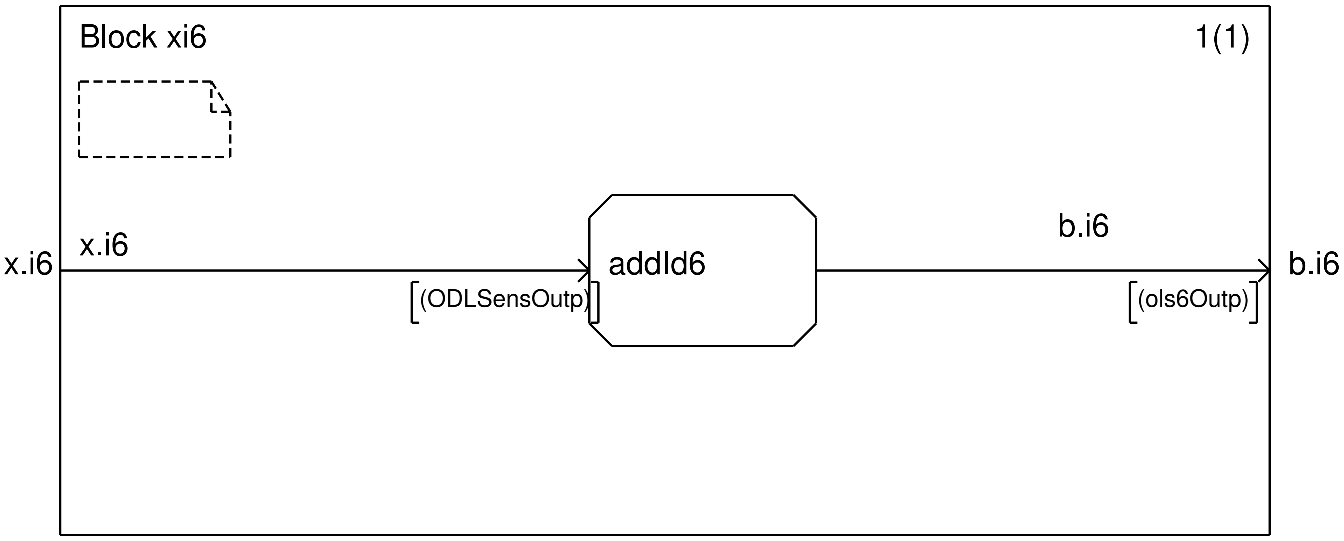

Block <<Substructure SunDet>> xi4
1(1)
x.i4
x.i4
(ODLSensOutp)
addId4
b.i4
b.i4
(ols4Outp)

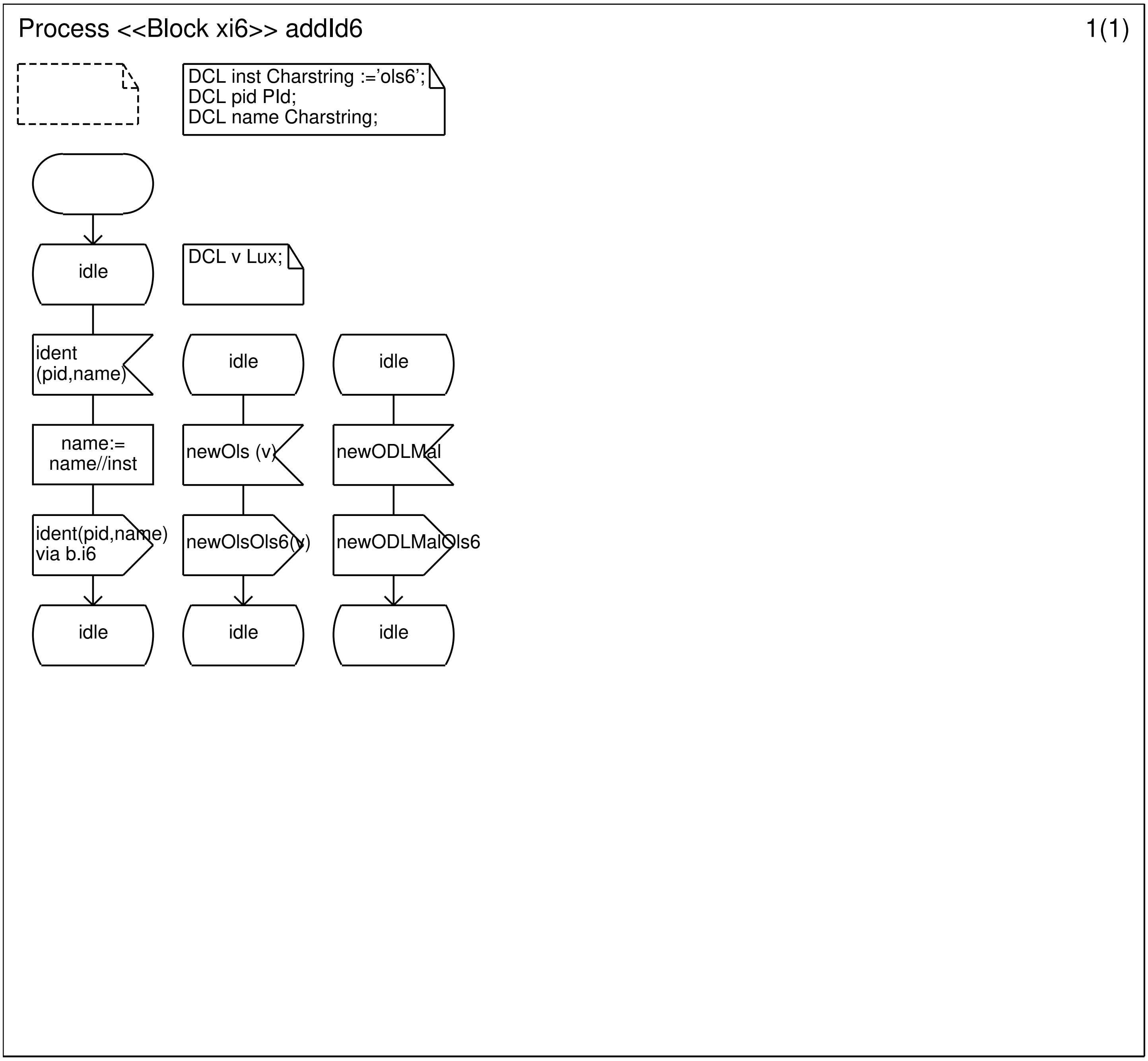


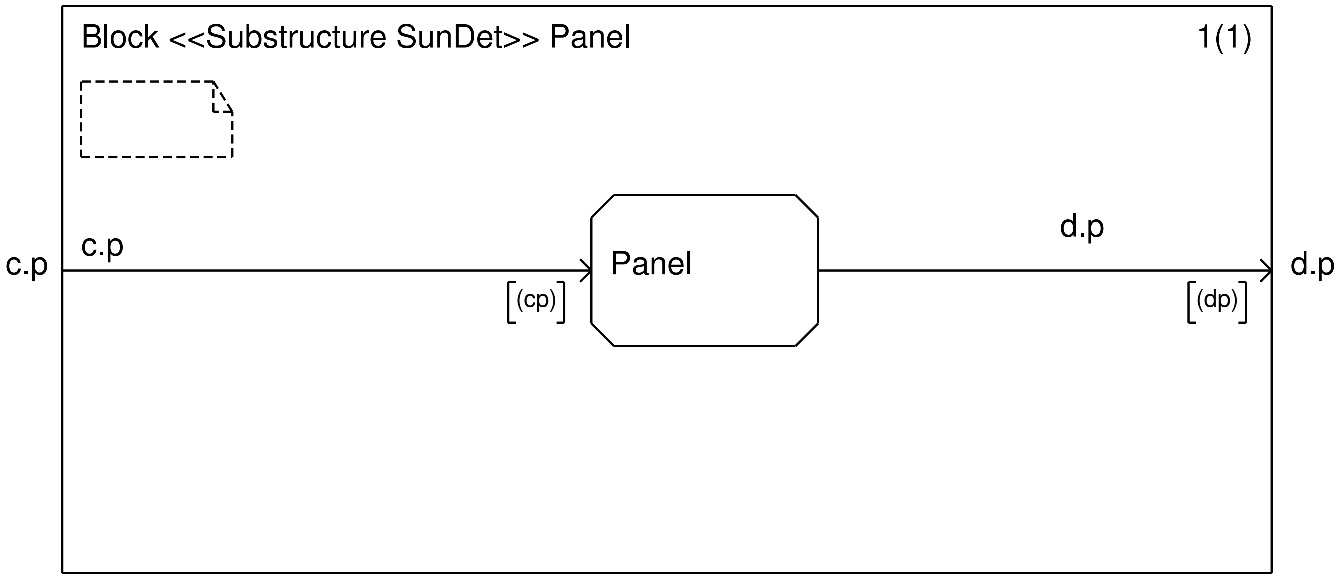
Process <<Substructure SunDet/Block xi4>> addId4
1(1)
DCL inst Charstring :='ols4';
DCL pid PId;
DCL name Charstring;
idle
DCL v Lux;
ident
(pid,name)
idle
idle
name:=
name//inst
newOls (v)
newODLMal
ident(pid,name)
via b.i4
newOlsOls4(v)
newODLMalOls4
idle
idle
idle


Block xi5
1(1)
x.i5
x.i5
(ODLSensOutp)
addId5
b.i5
b.i5
(ols5Outp)

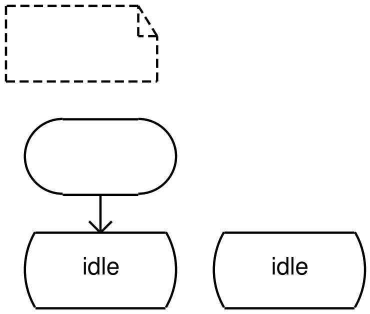

Process <<Block xi5>> addId5
1(1)
DCL inst Charstring :='ols5';
DCL pid PId;
DCL name Charstring;
idle
DCL v Lux;
ident
(pid,name)
idle
idle
name:=
name//inst
newOls (v)
newODLMal
ident(pid,name)
via b.i5
newOlsOls5(v)
newODLMalOls5
idle
idle
idle

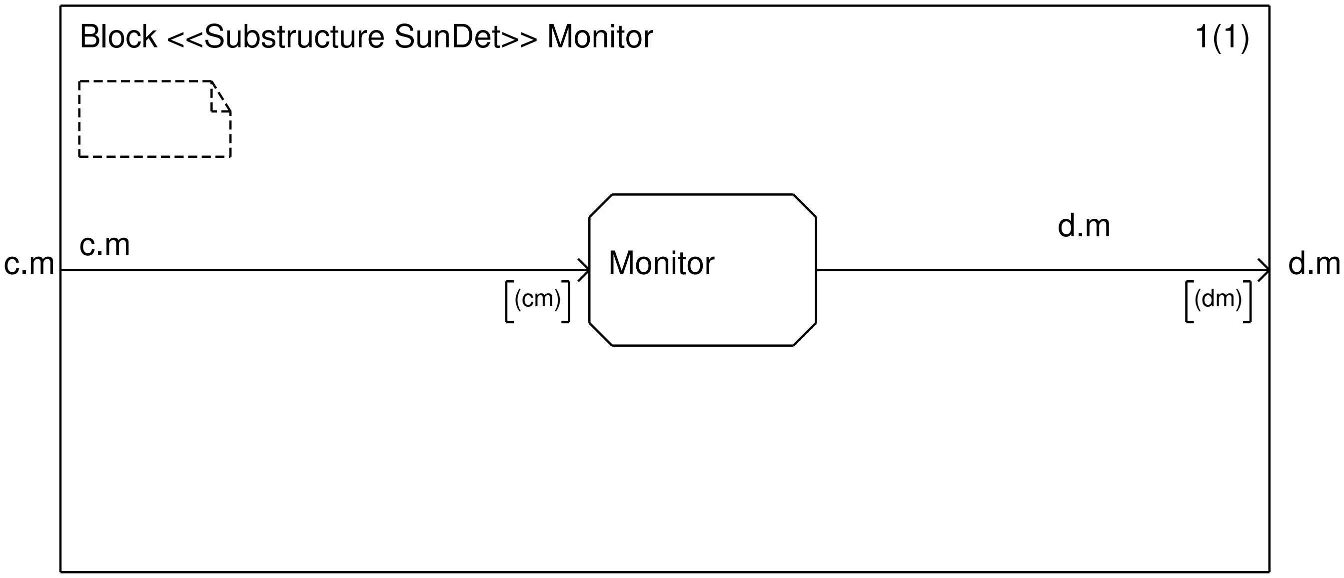

Block xi6
1(1)
x.i6
x.i6
(ODLSensOutp)
addId6
b.i6
b.i6
(ols6Outp)

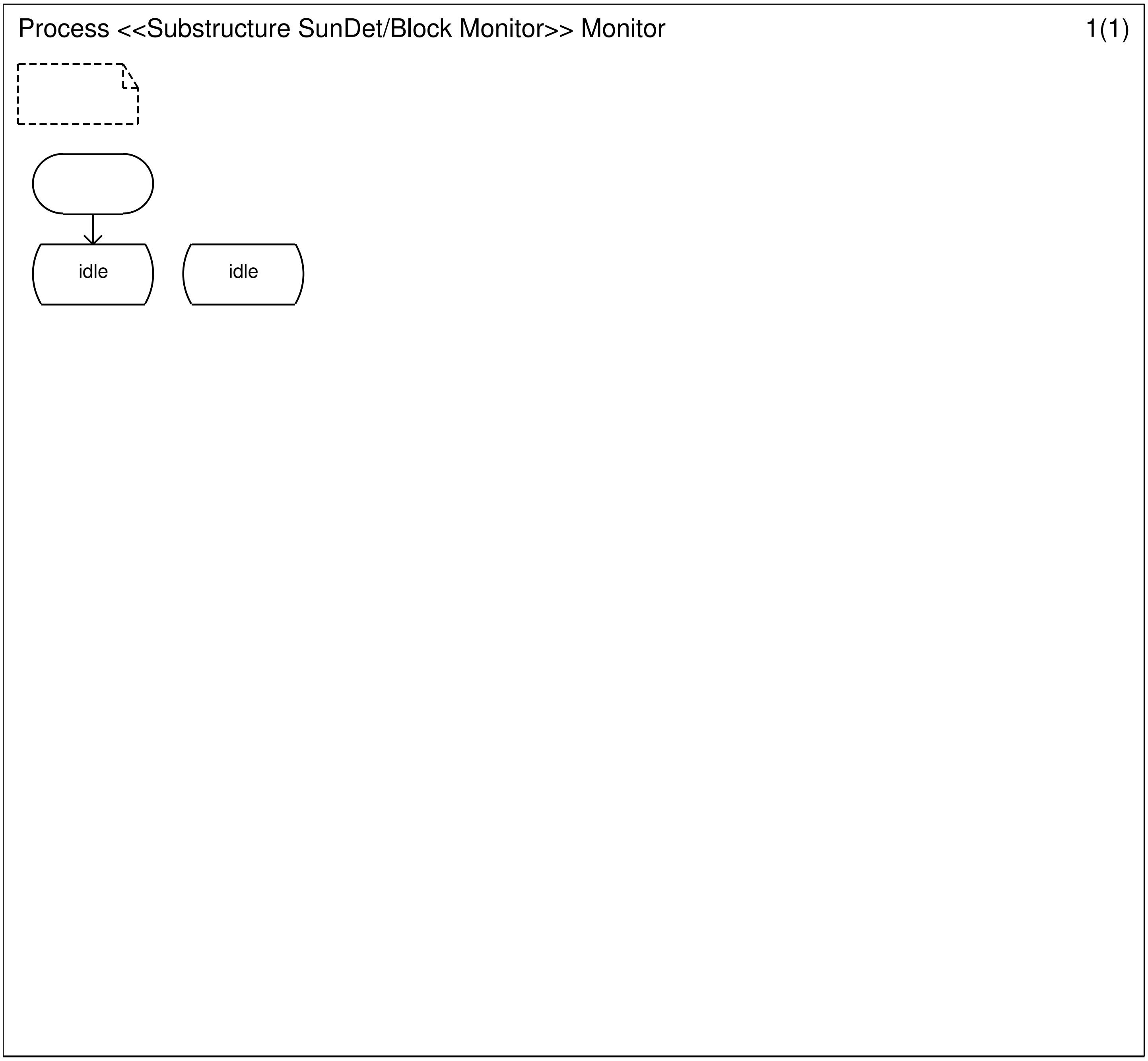

Process <<Block xi6>> addId6
1(1)
DCL inst Charstring :='ols6';
DCL pid PId;
DCL name Charstring;
idle
DCL v Lux;
ident
(pid,name)
name:=
name//inst
ident(pid,name)
via b.i6
idle
idle
newOls (v)
newOlsOls6(v)
idle
idle
newODLMal
newODLMalOls6
idle


Block <<Substructure SunDet>> Panel
1(1)
c.p
c.p
(cp)
Panel
d.p
d.p
(dp)

Process <<Substructure SunDet/Block Panel>> Panel 1(1)




Block <<Substructure SunDet>> Monitor
1(1)
c.m
c.m
(cm)
Monitor
d.m
d.m
(dm)


Process <<Substructure SunDet/Block Monitor>> Monitor
1(1)
idle
idle