Table of Contents

Organizer View 1
Package DataTypes 5
Package Utils 7
Package Sensors 9
Block Type ODLSens 10
Block Type MotDet 11
Block Type Switch 12
Block Type SunDet 13
Package Actuators 15
Block Type TaskLight 16
Block Type Dimmer 17
Block Type PulsGen 18
Package CP4 19
Block Type Floor 20
Package CP1 23
Block Type Section1 24
Block Type Section2 27
Block Type Section3 30
Package CP3 32
Block Type Hallway 33
Package CP2 36
System ProtoSys 41
Block ProtoCtrl 43
Process ProtoCtrl 44
Procedure putAnalogVal 49
Procedure putBinaryVal 50
Procedure pollPhysVal 51

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

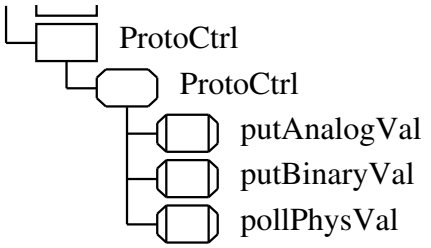
rw /SFB_501/teams/team3/Basis−Projekt.BaX/Entwicklung/RequirementsSp

Used Packages

DataTypes	r−	../Datatypes/LatestRev/DataTypes.sun
Utils	r−	../Utils/LatestRev/Utils.sun
Sensors	r−	../Sensors/Sensors.sun
ODLSens	r−	../Sensors/ODLSens.sbt
MotDet	r−	../Sensors/MotDet.sbt
Switch	r−	../Sensors/Switch.sbt
SunDet	r−	../Sensors/SunDet.sbt
Actuators	r−	../Actuators/LatestRev/Actuators.sun
TaskLight	r−	../Actuators/LatestRev/TaskLight.sbt
Dimmer	r−	../Actuators/LatestRev/Dimmer.sbt
PulsGen	r−	../Actuators/LatestRev/PulsGen.sbt
CP4	r−	../Control−sub4/LatestRev/CP4.sun
Floor	r−	../Control−sub4/LatestRev/Floor.sbt
CP1	r−	../Control−sub1/CP1.sun
Section1	r−	../Control−sub1/Section1.sbt
Section2	r−	../Control−sub1/Section2.sbt
Section3	r−	../Control−sub1/Section3.sbt
CP3	r−	../Control−sub3/LatestRev/CP3.sun
Hallway	r−	../Control−sub3/LatestRev/Hallway1.sbt
CP2	r−	../Control−sub2/CP2.sun

System

ProtoSys	rw	ProtoSys.ssy
x:y floor4 : Floor		



ProtoCtrl	rw	ProtoCtrl.sbk
ProtoCtrl	rw	ProtoCtrl.spr
putAnalogVal	rw	putAnalogVal.spd
putBinaryVal	rw	putBinaryVal.spd
pollPhysVal	rw	pollPhysVal.spd

ecification/System/ProtoSys.sdt

ecification/System/

Package Datatypes 1(2)

```
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 2(2)

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

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

```
USE DataTypes;
```

Package Utils 2(2)

```
/* 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

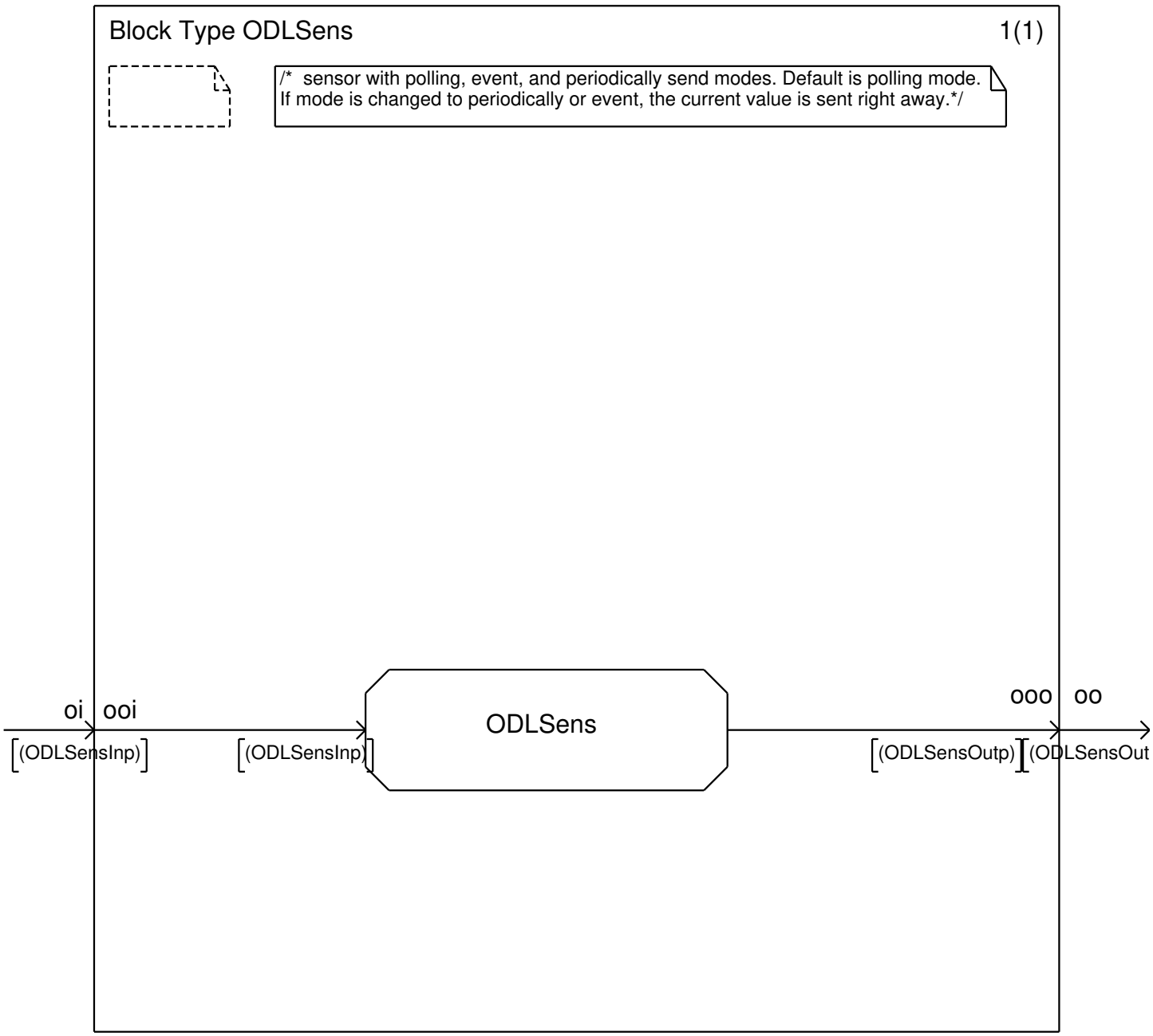
```
/* Signals at object input oi */
SIGNAL getMotDet ;  /*  Get value, independent of mode  */
SIGNAL setMDPeriod(Duration) ; /* Send value periodically with period p  */
SIGNAL setMDEvent ; /* Send value if changed  */
SIGNAL setMDPoll ; /* Wait for getValue signal  */
SIGNALLIST  MotDetInp=getMotDet,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

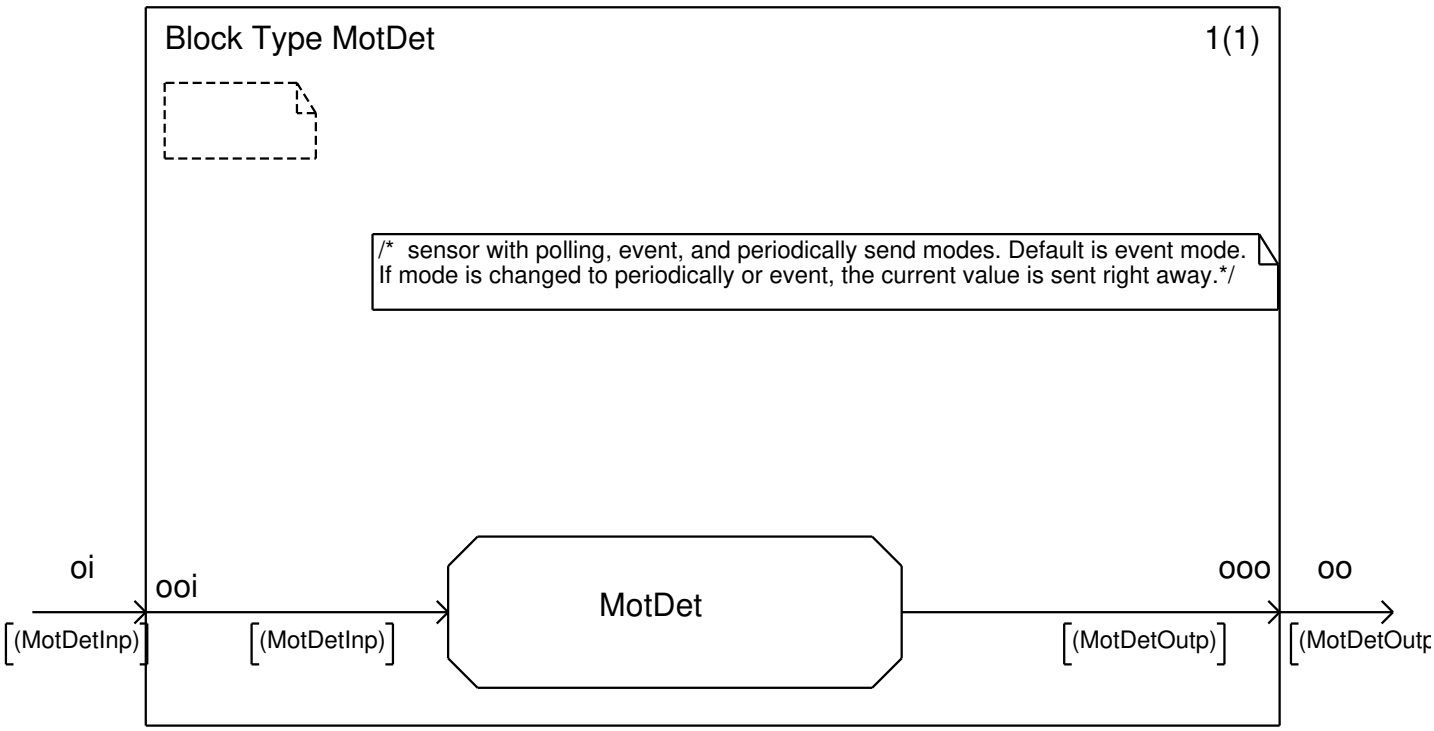
Switch

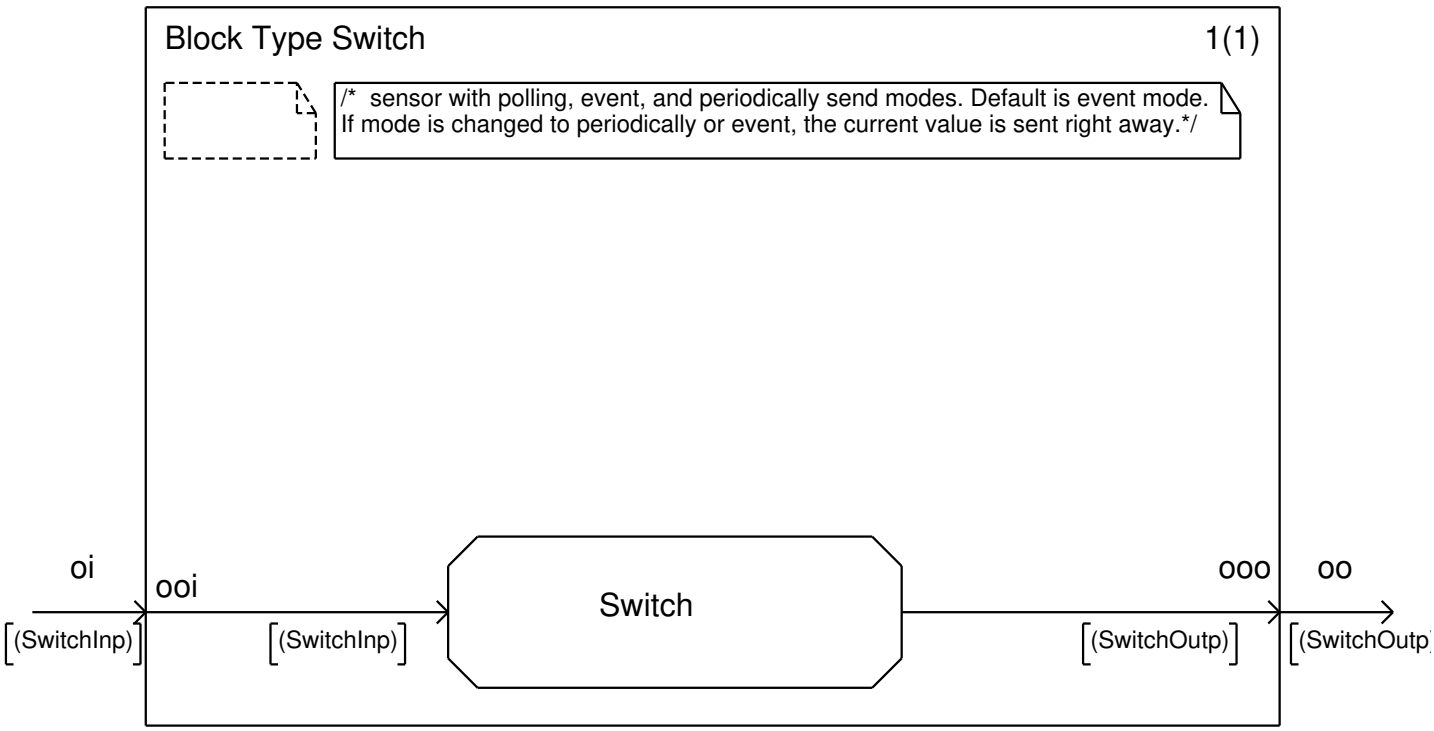
```
/* Signals at object input oi */
SIGNAL getSwitch ;  /*  Get value, independent of mode  */
SIGNAL setSwPeriod(Duration) ; /* Send value periodically with period p  */
SIGNAL setSwEvent ; /* Send value if changed  */
SIGNAL setSwPoll ; /* Wait for getValue signal  */
SIGNALLIST  SwitchInp=getSwitch,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

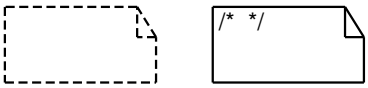
SunDet

```
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

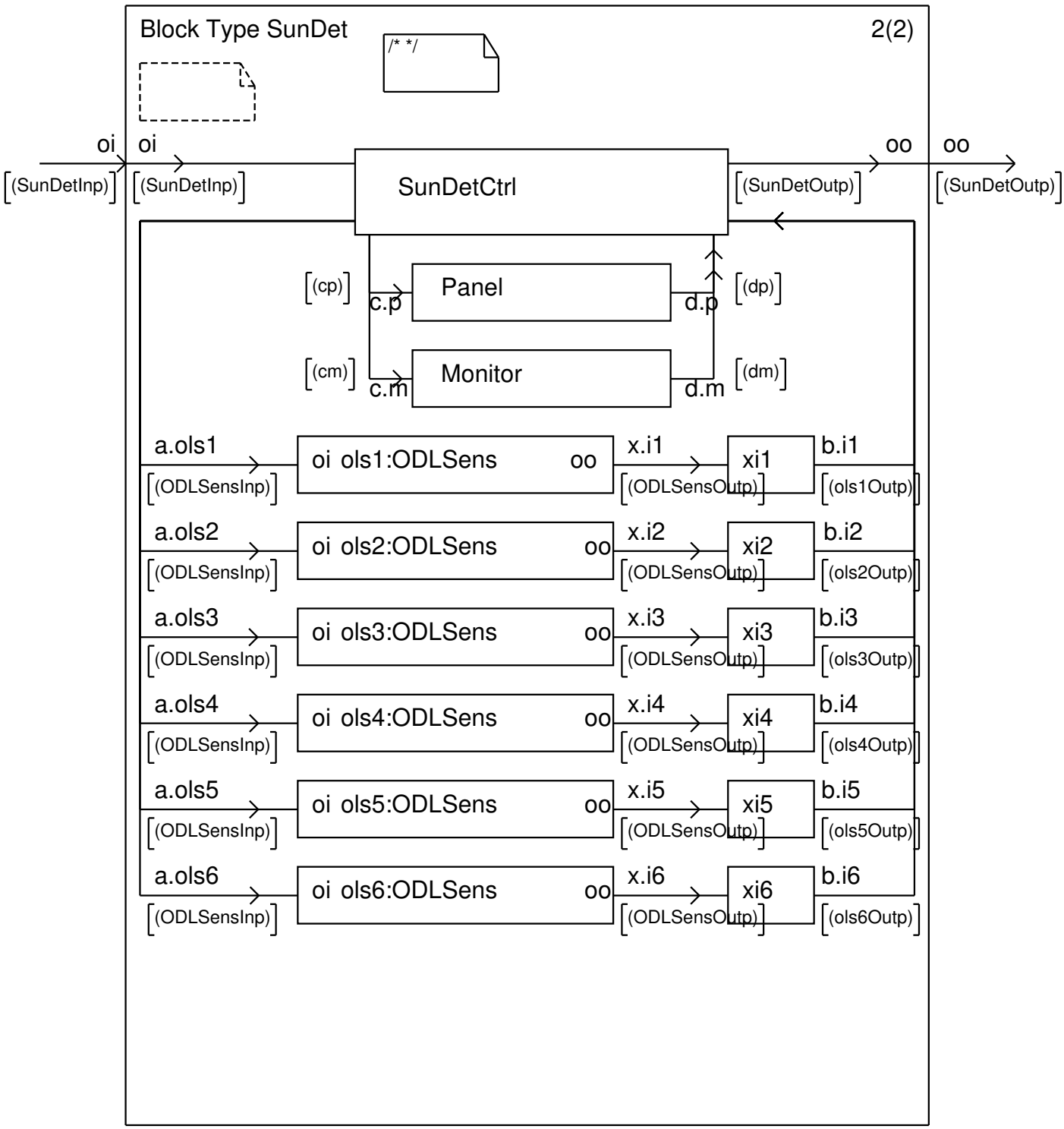

Block Type MotDet
1(1)
/* sensor with polling, event, and periodically send modes. Default is event mode.
If mode is changed to periodically or event, the current value is sent right away.*/
oi
ooi
MotDet
ooo
oo
(MotDetInp)
(MotDetInp)
(MotDetOutp)
(MotDetOutp


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
Switch
ooo
oo
(SwitchInp)
(SwitchInp)
(SwitchOutp)
(SwitchOutp)




```
/* 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)

```
USE DataTypes;
USE Utils;
```

Package Actuators 1(1)

TaskLight

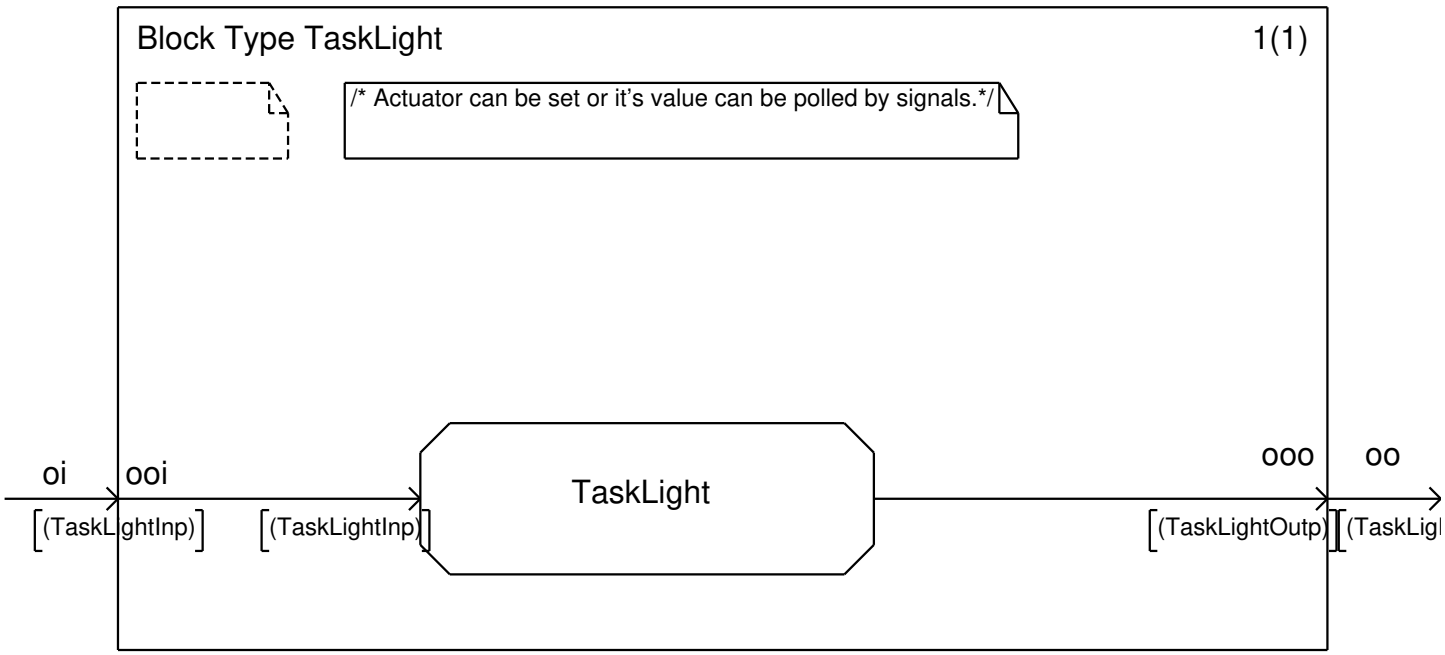
```
/* Signals at object input oi */
SIGNAL setTL(Binary);
SIGNALLIST  TaskLightInp= setTL; /*   */
SIGNALLIST TaskLightOutp=ident;
```

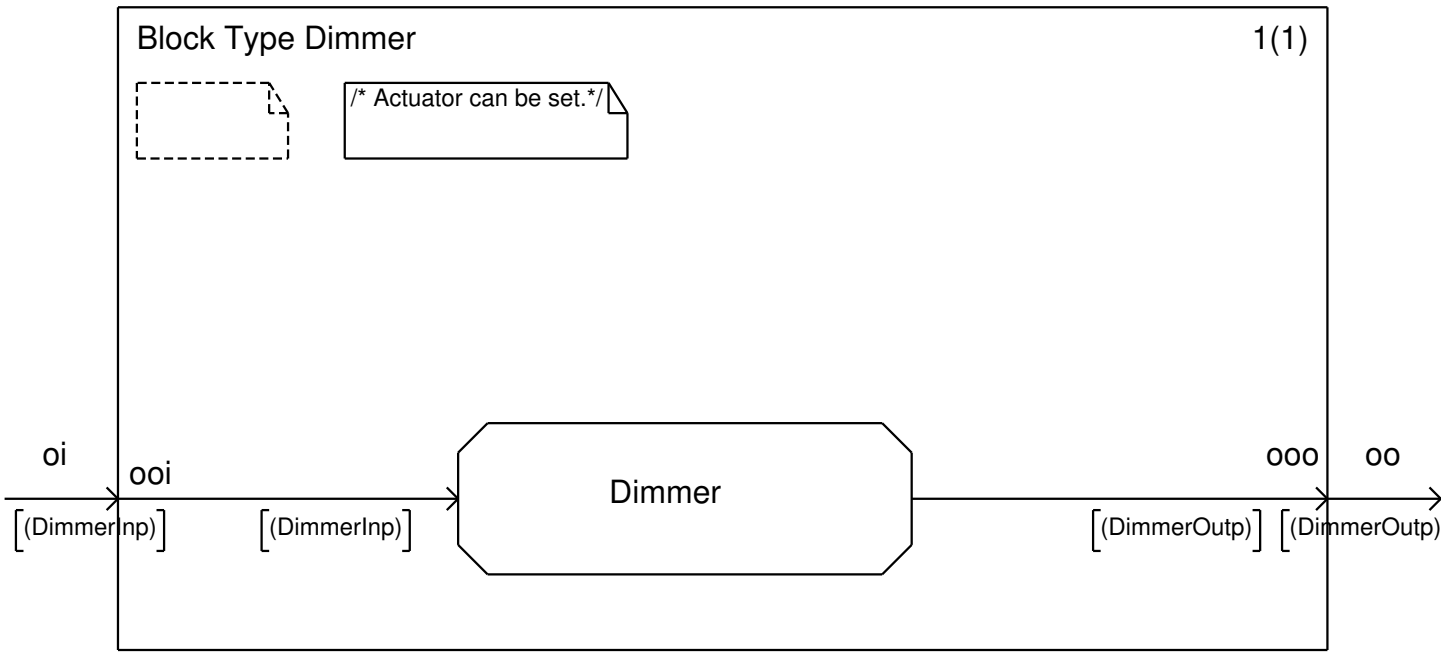
Dimmer

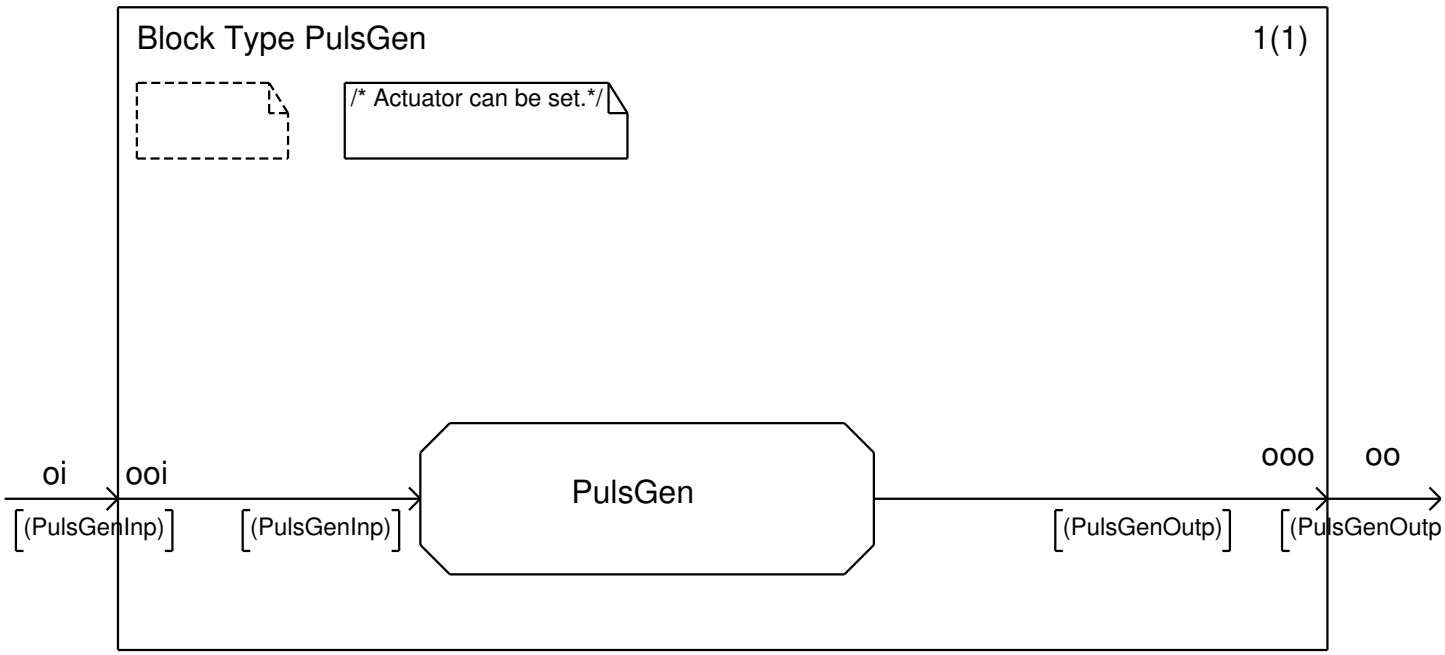
```
/* Signals at object input oi */
SIGNAL setDV(Percent);
SIGNALLIST  DimmerInp= setDV; /*   */
SIGNALLIST DimmerOutp=ident;
```

PulsGen

```
/* Signals at object input oi */
SIGNAL setPG(Binary);
SIGNALLIST  PulsGenInp= setPG; /*   */
SIGNALLIST PulsGenOutp=ident;
```


Block Type TaskLight
1(1)
/* Actuator can be set or it's value can be polled by signals.*/
oi
ooi
TaskLight
ooo
oo
(TaskLightInp)
(TaskLightInp)
(TaskLightOutp)


Block Type Dimmer
1(1)
/* Actuator can be set.*/
oi
ooi
Dimmer
ooo
oo
(DimmerInp)
(DimmerInp)
(DimmerOutp)
(DimmerOutp)


Block Type PulsGen
1(1)
/* Actuator can be set.*/
oi
ooi
PulsGen
ooo
oo
(PulsGenInp)
(PulsGenInp)
(PulsGenOutp)
(PulsGenOutp

USE DataTypes;
USE Utils;
USE Sensors;
USE Actuators; USE CP1; USE CP2; USE CP3;

Package CP4 1(1)

Floor

SIGNALLIST FloorOutp = ident;
SIGNALLIST FloorInp = dummy;

Block Type Floor 1(3)

/* */

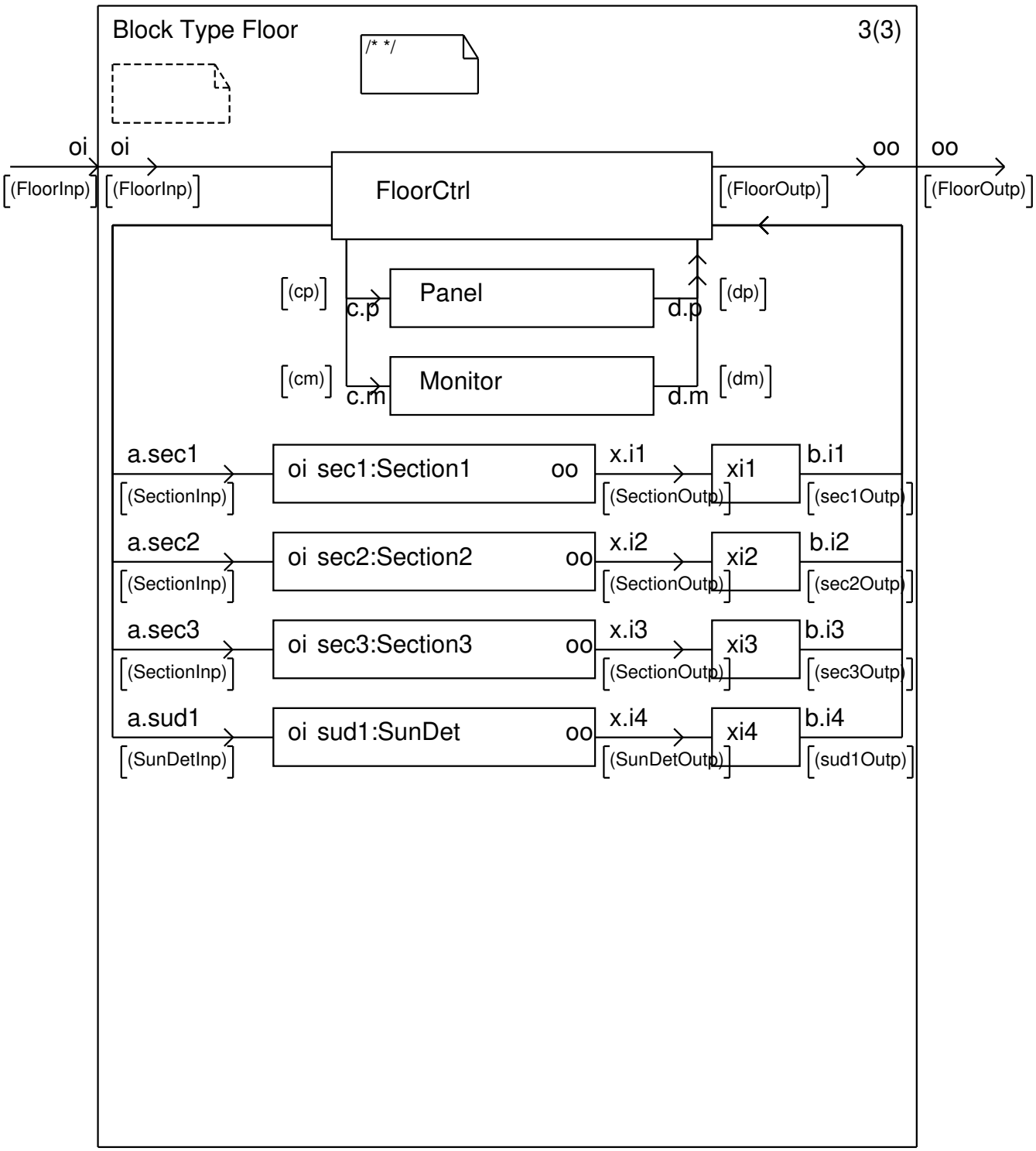
```
/* Signals to and from panel and monitor */
SIGNALLIST cp = dummy ; /*   */
SIGNALLIST cm = dummy ; /*   */
SIGNALLIST dp = dummy ; /*   */
SIGNALLIST dm = dummy ; /*   */
```

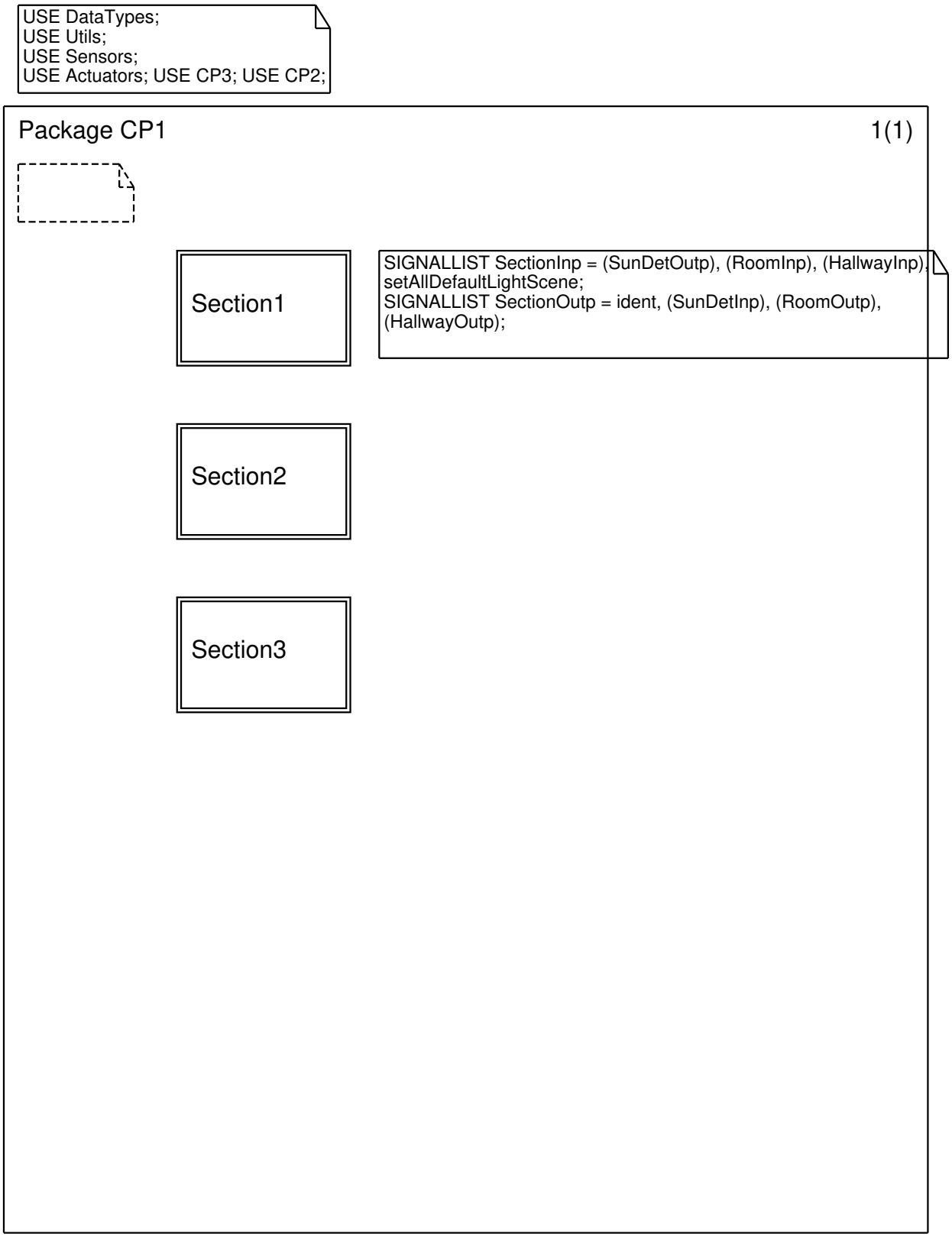
```
SIGNAL newSunDetMalSud1 (ErrorType);
SIGNAL newSunSud1 (RoomIdType, Lux);
SIGNALLIST sud1Outp = ident,
newSunDetMalSud1, newSunSud1 ; /*   */
```

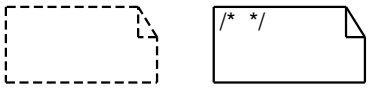
```
/* Signals from subobjects via b.sec1 */
SIGNAL newRoomMalFunctionSec1 (RoomIdType, ErrorType);
SIGNAL newRoomEnergyConsumptionSec1 (RoomIdType, Joule, Duration);
SIGNAL newHwEnergyConsumptionSec1(Joule, Duration);
SIGNAL newHwMalFunctionSec1 (ErrorType);
SIGNAL getHwPbDelaySec1;
SIGNAL getSunSec1 (RoomIdType);
SIGNALLIST sec1Outp = ident,
newRoomMalFunctionSec1, newRoomEnergyConsumptionSec1,
newHwEnergyConsumptionSec1,
newHwMalFunctionSec1, getHwPbDelaySec1, getSunSec1; /*   */
```

```
/* Signals from subobjects via b.sec2 */
SIGNAL newRoomMalFunctionSec2 (RoomIdType, ErrorType);
SIGNAL newRoomEnergyConsumptionSec2 (RoomIdType, Joule, Duration);
SIGNAL newHwEnergyConsumptionSec2(Joule, Duration);
SIGNAL newHwMalFunctionSec2 (ErrorType);
SIGNAL getHwPbDelaySec2;
SIGNAL getSunSec2 (RoomIdType);
SIGNALLIST sec2Outp = ident, getSunSec2,
newRoomMalFunctionSec2, newRoomEnergyConsumptionSec2,
newHwEnergyConsumptionSec2,
newHwMalFunctionSec2, getHwPbDelaySec2; /*   */
```

```
/* Signals from subobjects via b.sec3 */
SIGNAL newRoomMalFunctionSec3 (RoomidType, ErrorType);
SIGNAL newRoomEnergyConsumptionSec3 (RoomIdType, Joule, Duration);
SIGNAL newHwEnergyConsumptionSec3(Joule, Duration);
SIGNAL newHwMalFunctionSec3 (ErrorType);
SIGNAL getHwPbDelaySec3;
SIGNAL getSunSec3 (RoomIdType);
SIGNALLIST sec3Outp = ident, getSunSec3,
newRoomMalFunctionSec3, newRoomEnergyConsumptionSec3,
newHwEnergyConsumptionSec3,
newHwMalFunctionSec3, getHwPbDelaySec3; /*   */
```

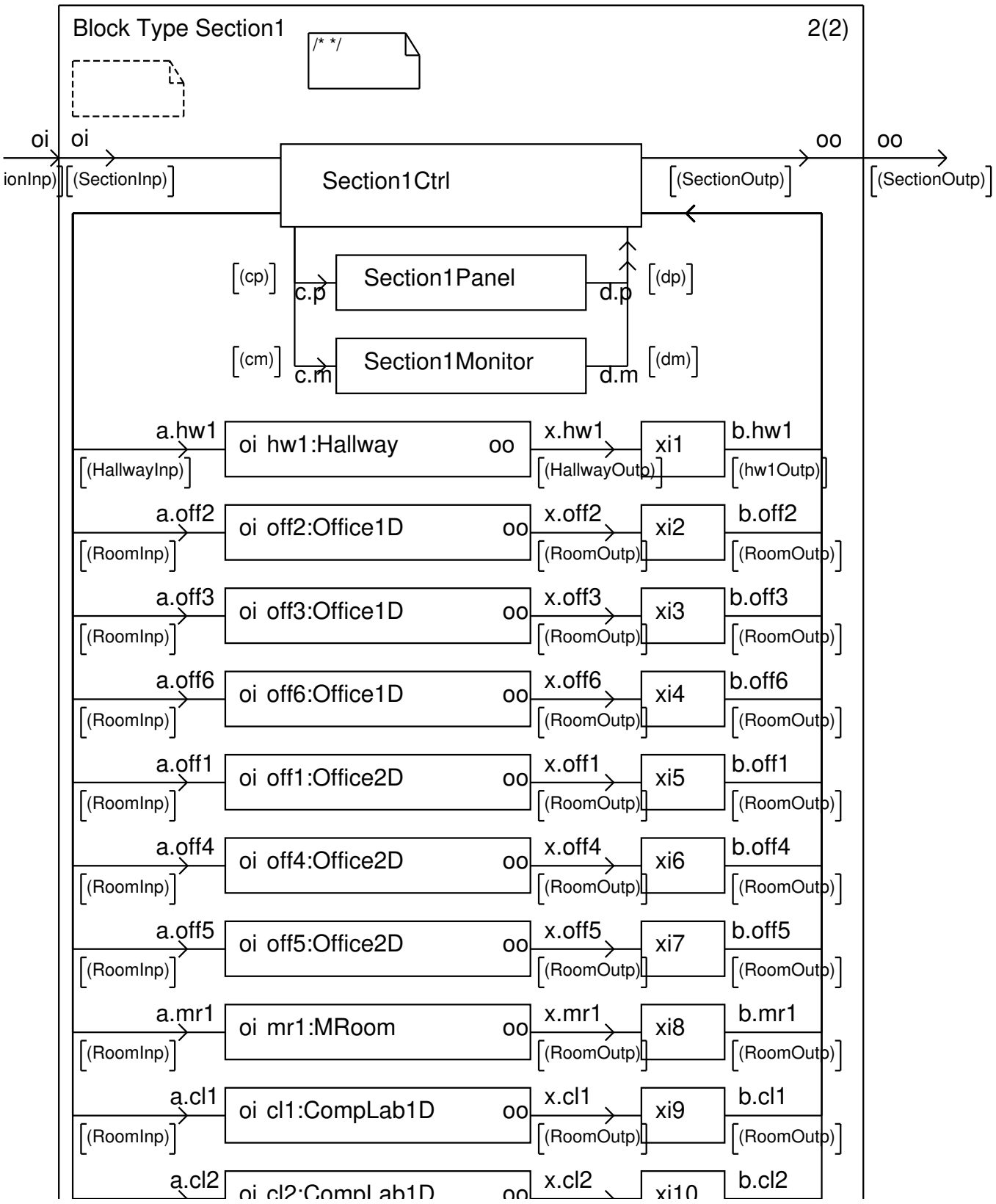

Block Type Floor
3(3)
/* */
oi
oi
(FloorInp)
(FloorInp)
FloorCtrl
oo
oo
(FloorOutp)
(FloorOutp)
(cp)
c.p
Panel
d.p
(dp)
(cm)
c.m
Monitor
d.m
(dm)
a.sec1
(SectionInp)
oi sec1:Section1 oo
x.i1
(SectionOutp)
xi1
b.i1
(sec1Outp)
a.sec2
(SectionInp)
oi sec2:Section2 oo
x.i2
(SectionOutp)
xi2
b.i2
(sec2Outp)
a.sec3
(SectionInp)
oi sec3:Section3 oo
x.i3
(SectionOutp)
xi3
b.i3
(sec3Outp)
a.sud1
(SunDetInp)
oi sud1:SunDet oo
x.i4
(SunDetOutp)
xi4
b.i4
(sud1Outp)

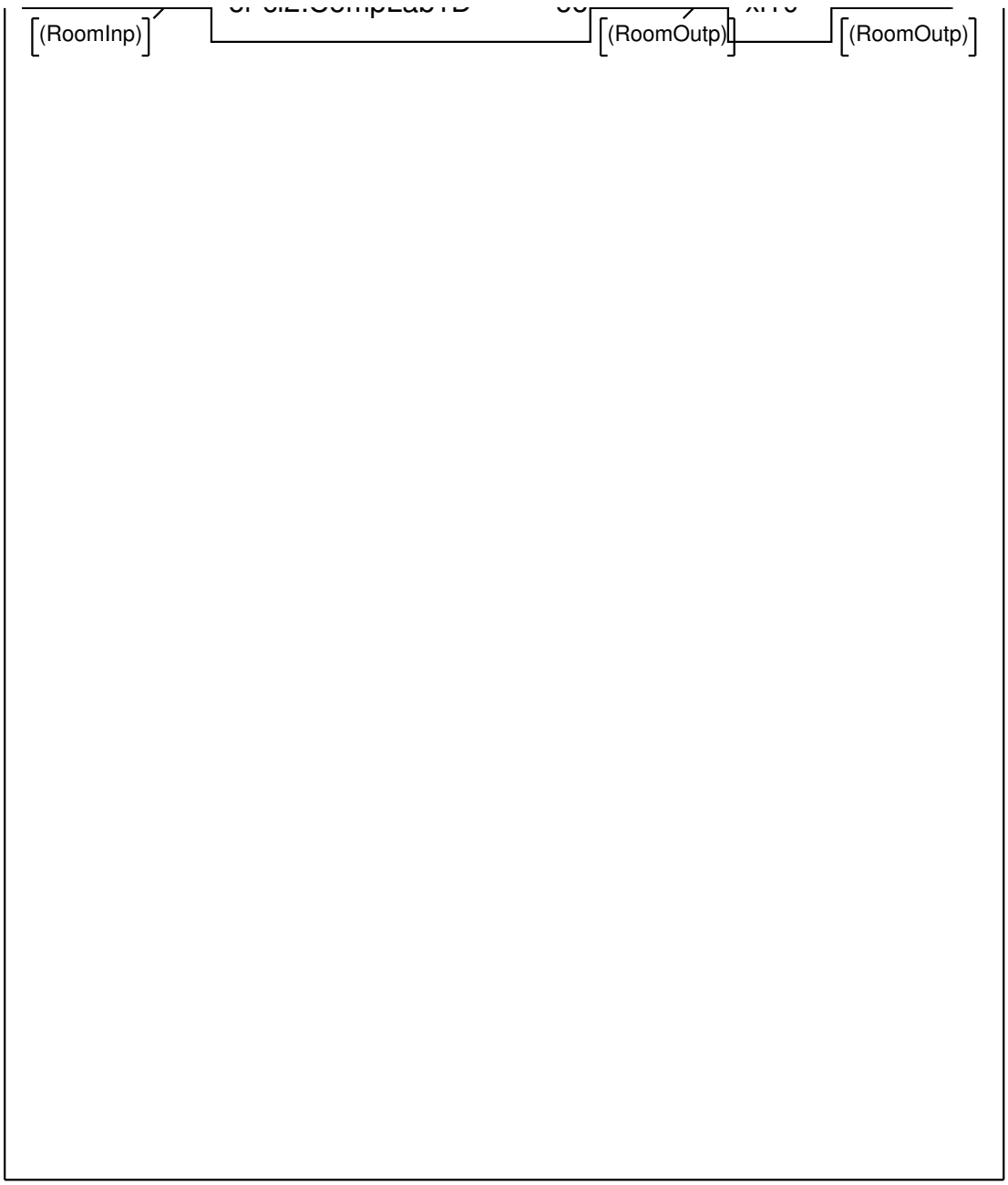

USE DataTypes;
USE Utils;
USE Sensors;
USE Actuators; USE CP3; USE CP2;
Package CP1
1(1)
Section1
SIGNALLIST SectionInp = (SunDetOutp), (RoomInp), (HallwayInp), setAllDefaultLightScene;
SIGNALLIST SectionOutp = ident, (SunDetInp), (RoomOutp), (HallwayOutp);
Section2
Section3

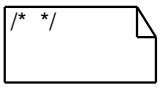



```
/* Signals to and from panel and monitor */
SIGNALLIST cp = dummy ; /*   */
SIGNALLIST cm = dummy ; /*   */
SIGNALLIST dp = dummy ; /*   */
SIGNALLIST dm = dummy ; /*   */
```

```
SIGNAL getIlluminanceHw1;
SIGNAL newHwEnergyConsumptionHw1( Joule, Duration);
SIGNAL newHwMalFunctionHw1( Errortype );
SIGNAL getHwPBDelayHw1;

SIGNALLIST hw1Outp =  ident, getIlluminanceHw1,
                      newHwEnergyConsumptionHw1,
                      newHwMalFunctionHw1, getHwPBDelayHw1;
```

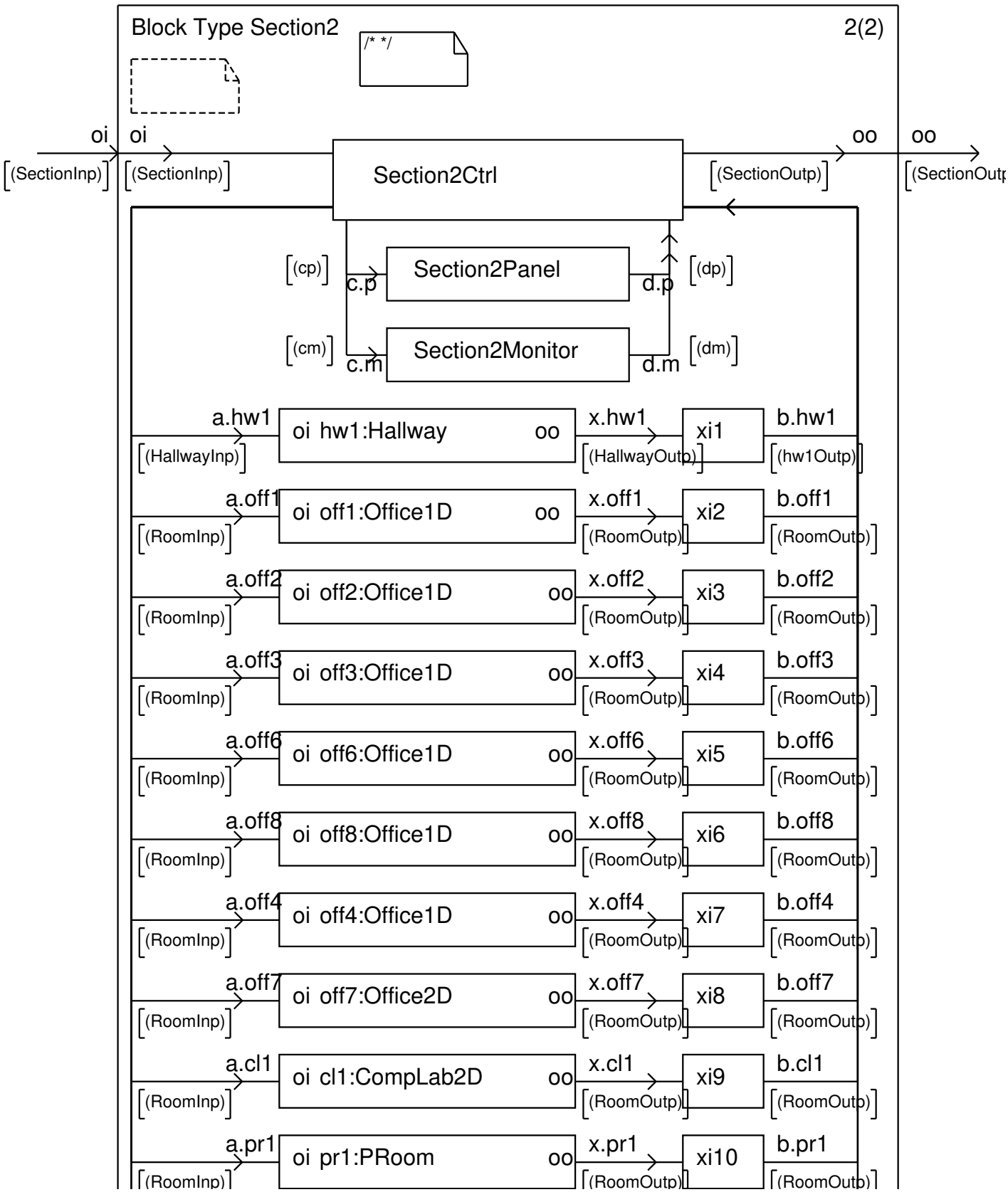

Block Type Section1
2(2)
/* */
oi
oi
oo
oo
ionInp)
(SectionInp)
Section1Ctrl
(SectionOutp)
(SectionOutp)
(cp)
c.p
Section1Panel
d.p
(dp)
(cm)
c.m
Section1Monitor
d.m
(dm)
a.hw1
oi hw1:Hallway oo
x.hw1
xi1
b.hw1
(HallwayInp)
(HallwayOutp)
(hw1Outp)
a.off2
oi off2:Office1D oo
x.off2
xi2
b.off2
(RoomInp)
(RoomOutp)
(RoomOutp)
a.off3
oi off3:Office1D oo
x.off3
xi3
b.off3
(RoomInp)
(RoomOutp)
(RoomOutp)
a.off6
oi off6:Office1D oo
x.off6
xi4
b.off6
(RoomInp)
(RoomOutp)
(RoomOutp)
a.off1
oi off1:Office2D oo
x.off1
xi5
b.off1
(RoomInp)
(RoomOutp)
(RoomOutp)
a.off4
oi off4:Office2D oo
x.off4
xi6
b.off4
(RoomInp)
(RoomOutp)
(RoomOutp)
a.off5
oi off5:Office2D oo
x.off5
xi7
b.off5
(RoomInp)
(RoomOutp)
(RoomOutp)
a.mr1
oi mr1:MRoom oo
x.mr1
xi8
b.mr1
(RoomInp)
(RoomOutp)
(RoomOutp)
a.cl1
oi cl1:CompLab1D oo
x.cl1
xi9
b.cl1
(RoomInp)
(RoomOutp)
(RoomOutp)
a.cl2
x.cl2
b.cl2

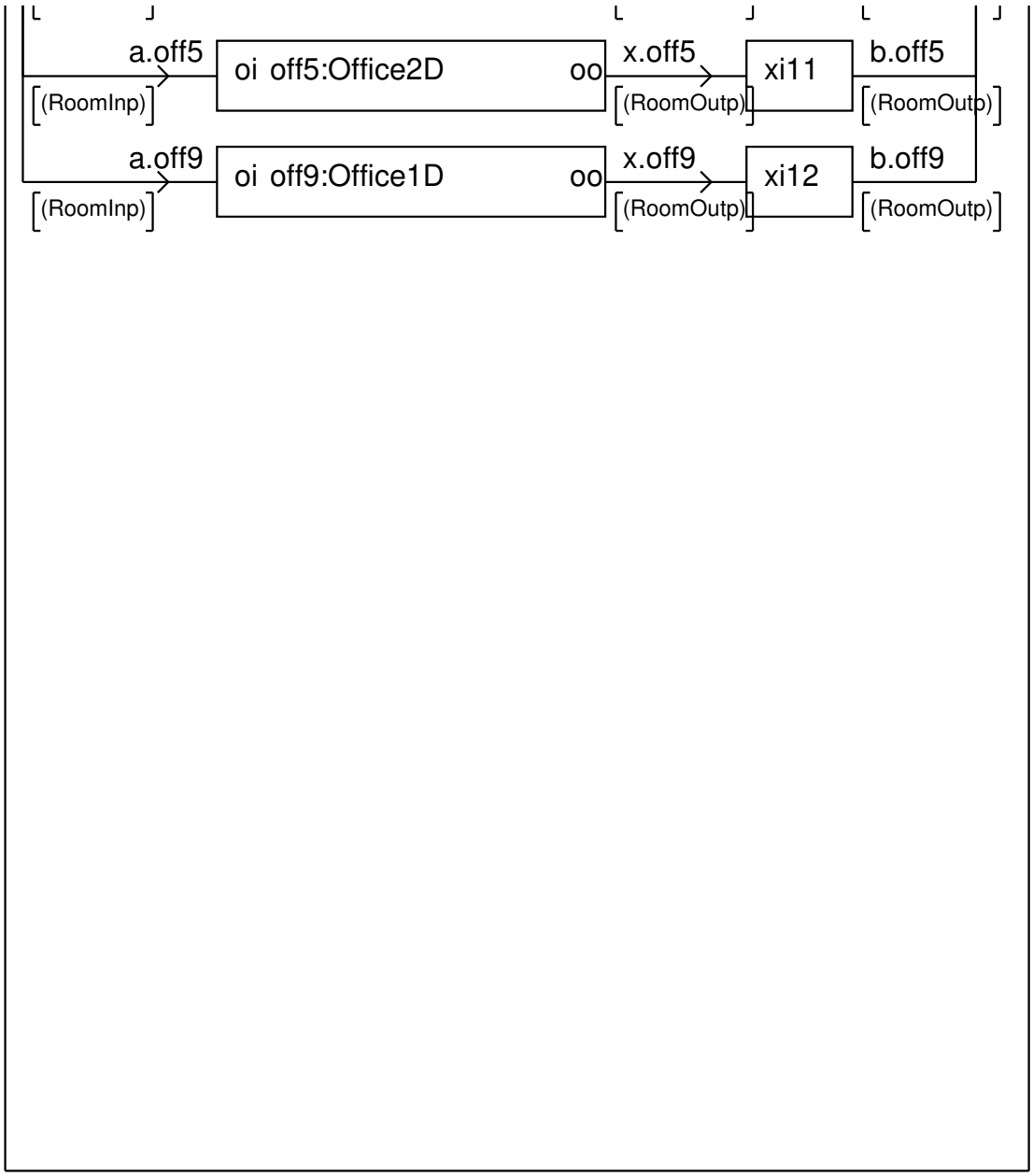

(RoomInp)
(RoomOutp)
(RoomOutp)

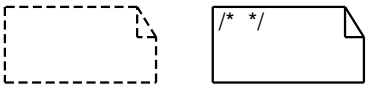



```
/* Signals to and from panel and monitor */
SIGNALLIST cp = dummy ; /*   */
SIGNALLIST cm = dummy ; /*   */
SIGNALLIST dp = dummy ; /*   */
SIGNALLIST dm = dummy ; /*   */
```

```
SIGNAL getIlluminanceHw1;
SIGNAL newHwEnergyConsumptionHw1( Joule, Duration);
SIGNAL newHwMalFunctionHw1( Errortype );
SIGNAL getHwPBDelayHw1;

SIGNALLIST hw1Outp =  ident, getIlluminanceHw1, newHwEnergyConsumptionHw1,
                newHwMalFunctionHw1, getHwPBDelayHw1;
```

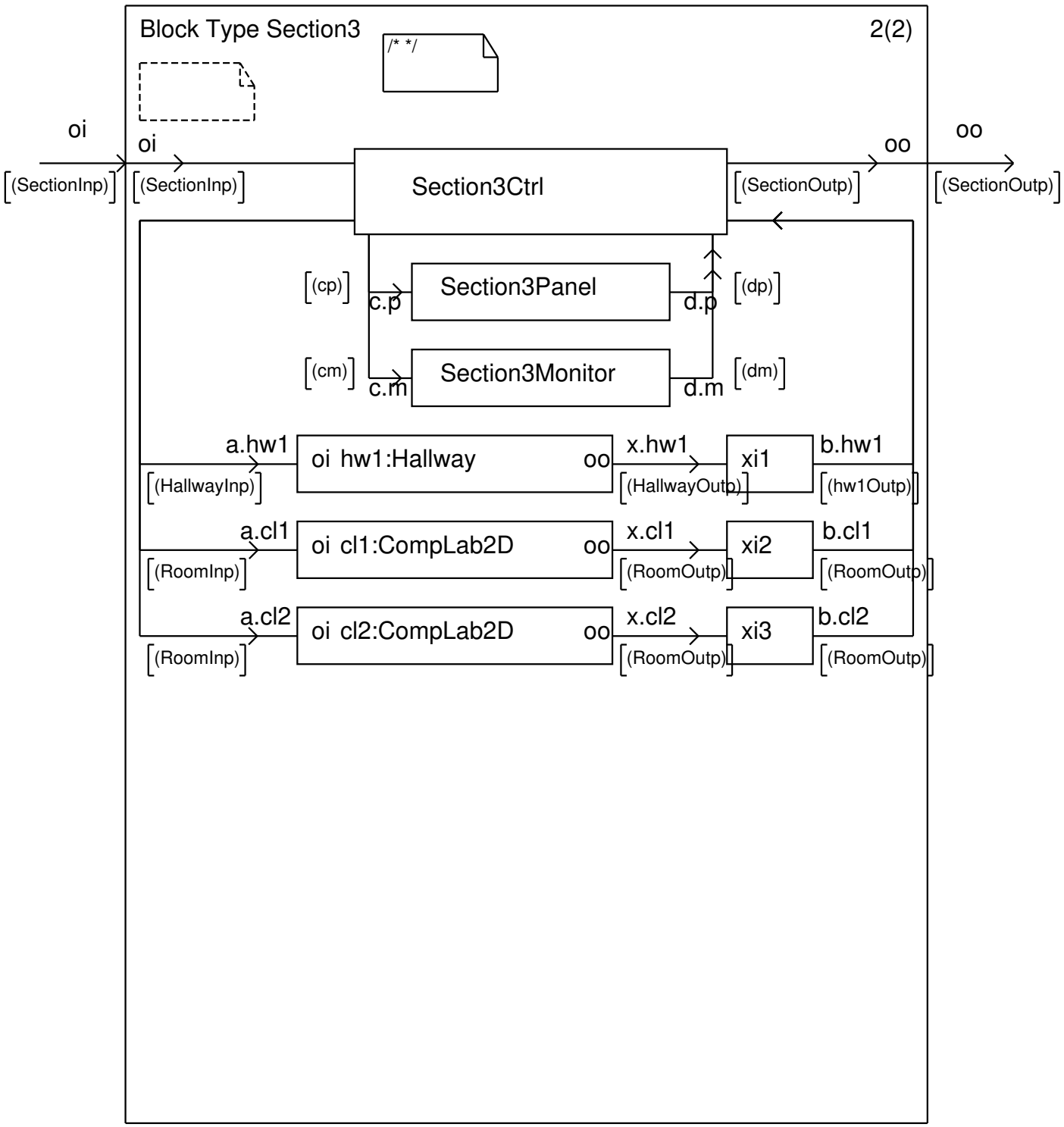

Block Type Section2
2(2)
/* */
oi
oi
oo
oo
(SectionInp)
(SectionInp)
Section2Ctrl
(SectionOutp)
(SectionOutp
(cp)
c.p
Section2Panel
d.p
(dp)
(cm)
c.m
Section2Monitor
d.m
(dm)
a.hw1
oi hw1:Hallway oo
x.hw1
xi1
b.hw1
(HallwayInp)
(HallwayOutp)
(hw1Outp)
a.off1
oi off1:Office1D oo
x.off1
xi2
b.off1
(RoomInp)
(RoomOutp)
(RoomOutp)
a.off2
oi off2:Office1D oo
x.off2
xi3
b.off2
a.off3
oi off3:Office1D oo
x.off3
xi4
b.off3
a.off6
oi off6:Office1D oo
x.off6
xi5
b.off6
a.off8
oi off8:Office1D oo
x.off8
xi6
b.off8
a.off4
oi off4:Office1D oo
x.off4
xi7
b.off4
a.off7
oi off7:Office2D oo
x.off7
xi8
b.off7
a.cl1
oi cl1:CompLab2D oo
x.cl1
xi9
b.cl1
a.pr1
oi pr1:PRoom oo
x.pr1
xi10
b.pr1


a.off5
oi off5:Office2D
oo
x.off5
xi11
b.off5
(RoomInp)
(RoomOutp)
(RoomOutp)
a.off9
oi off9:Office1D
oo
x.off9
xi12
b.off9
(RoomInp)
(RoomOutp)
(RoomOutp)




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

```
SIGNAL getIlluminanceHw1;
SIGNAL newHwEnergyConsumptionHw1( Joule, Duration);
SIGNAL newHwMalFunctionHw1( Errortype );
SIGNAL getHwPBDelayHw1;

SIGNALLIST hw1Outp =  ident, getIlluminanceHw1, newHwEnergyConsumptionHw1,
                newHwMalFunctionHw1, getHwPBDelayHw1;
```


Block Type Section3
2(2)
/* */
oi
oi
oo
oo
(SectionInp)
(SectionInp)
Section3Ctrl
(SectionOutp)
(SectionOutp)
(cp)
c.p
Section3Panel
d.p
(dp)
(cm)
c.m
Section3Monitor
d.m
(dm)
a.hw1
oi hw1:Hallway oo
x.hw1
xi1
b.hw1
(HallwayInp)
(HallwayOutp)
(hw1Outp)
a.cl1
oi cl1:CompLab2D oo
x.cl1
xi2
b.cl1
(RoomInp)
(RoomOutp)
(RoomOutp)
a.cl2
oi cl2:CompLab2D oo
x.cl2
xi3
b.cl2
(RoomInp)
(RoomOutp)
(RoomOutp)

```
USE DataTypes;
USE Utils;
USE Sensors;
USE Actuators;
```

Package CP3 1(1)

Hallway

```
SIGNAL setHwLight2;

SIGNALLIST HallwayOutp = ident, getIlluminance,
                newHwEnergyConsumption,
                newHwMalFunction, getHwPBDelay, setHwLight2;


SIGNALLIST HallwayInp = setHwEnergyDiffTime, setHwLight,
                 setHwPbDelay, newIlluminance,
                 /*newSll,*/ newEntrance, setHWOccDelay;
```

Block Type Hallway 1(3)

HWLight

```
SIGNAL newEnergy (Joule);
SIGNAL newSll( Binary );
SIGNALLIST HWLightOutp = ident, newSll, newEnergy;
SIGNAL setEnergyDiffTime (Duration);
SIGNAL setHwLight2( Boolean );
SIGNALLIST HWLightInp = setEnergyDiffTime, setHwLight2;
```

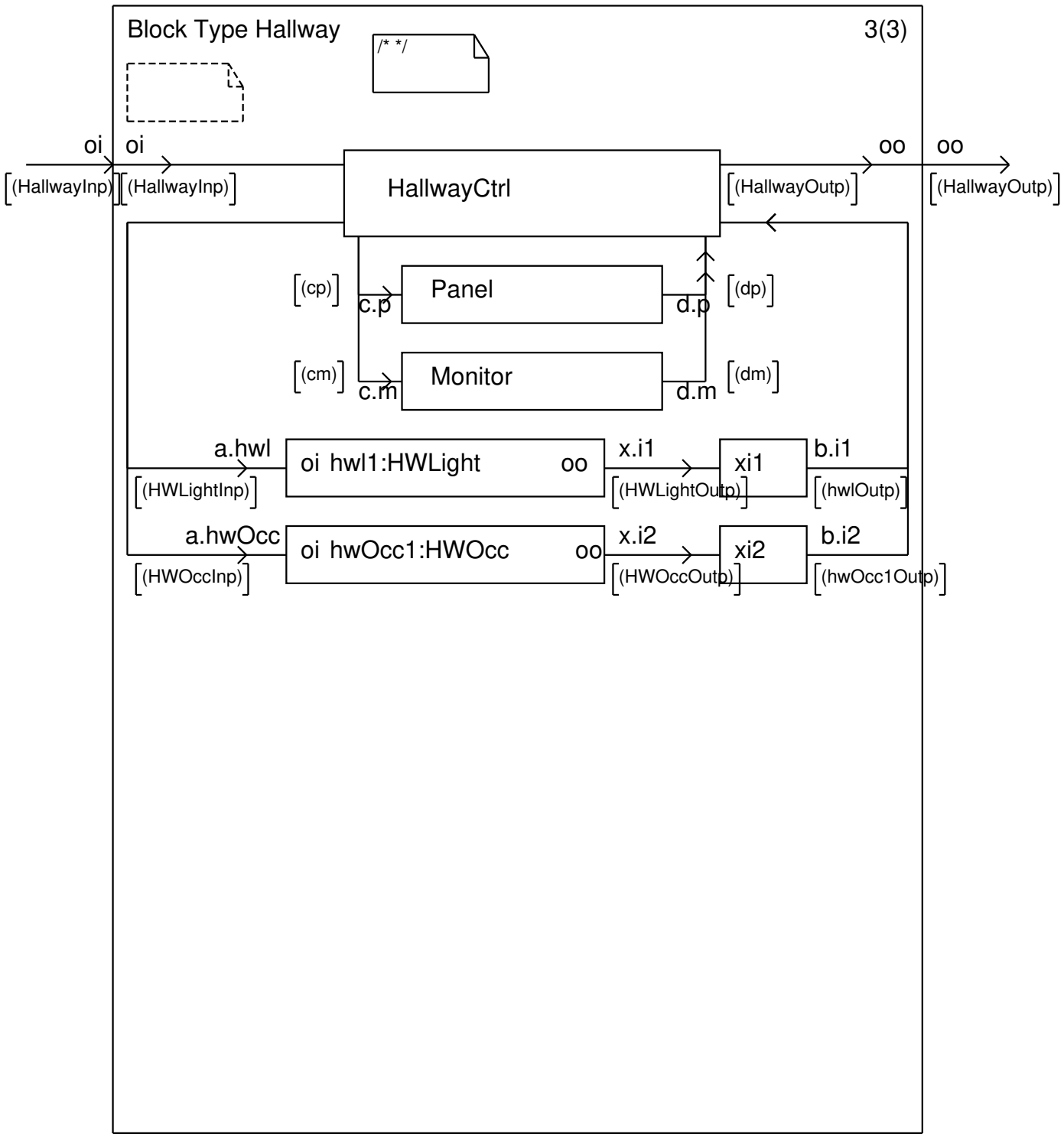
HWOcc

```
SIGNAL newHwOcc( Boolean );
SIGNALLIST HWOccOutp = ident, newHwOcc, getHwOccDelay;

SIGNAL getHwOcc;
SIGNALLIST HWOccInp = newEntrance, getHwOcc,
                      setHWOccDelay;
```

```
/*  */
```

```
/* 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 newSllHwl1 (Binary);
SIGNAL newEnergyHwl1 (Joule);
SIGNALLIST hwlOutp = ident, newEnergyHwl1, newSllHwl1 ; /*   */

SIGNAL newHwOccHwOcc1(Boolean);
SIGNAL getHwOccDelayHwOcc1;
SIGNALLIST hwOcc1Outp = ident, newHwOccHwOcc1,
   getHwOccDelayHwOcc1 ; /*   */
```


Block Type Hallway
3(3)
/* */
oi
oi
(HallwayInp)
(HallwayInp)
HallwayCtrl
oo
oo
(HallwayOutp)
(HallwayOutp)
(cp)
c.p
Panel
d.p
(dp)
(cm)
c.m
Monitor
d.m
(dm)
a.hwl
oi hwl1:HWLight oo
x.i1
xi1
b.i1
(HWLightInp)
(HWLightOutp)
(hwlOutp)
a.hwOcc
oi hwOcc1:HWOcc oo
x.i2
xi2
b.i2
(HWOccInp)
(HWOccOutp)
(hwOcc1Outp)

```
USE DataTypes;
USE Utils;
USE Sensors;
USE Actuators;
```

Package CP2

```
/* Signal Declarations */


/* ---- Door ----*/
SIGNAL newDoor (Boolean);
SIGNAL getDoor;
/* ---- RoomOcc ---- */
SIGNAL newOcc (Boolean);
SIGNAL newOccMal (ErrorType);
SIGNAL newDoorsOpen (Boolean);
/* ---- Desk ---- */
SIGNAL newTaskLight (Boolean);
SIGNAL setTaskLight (Boolean);
SIGNAL getTaskLight;
/* ---- Rooms ---- */
SIGNAL setDimLight (Percent);
SIGNAL setLight (Boolean);
SIGNAL getLight;
SIGNAL getDimLight;
SIGNAL setCIA;
SIGNAL getOcc;
SIGNAL newLight (Boolean);
SIGNAL newEnergy (Joule);
SIGNAL setEnergyDiffTime (Duration);

/* ---- Light ----*/
```

USE DataTypes;
USE Utils;
USE Sensors;
USE Actuators;

Package CP2 2(5)

SIGNALLIST DoorOutp = ident, newDoor;
SIGNALLIST DoorInp = getDoor;

SIGNALLIST RoomOccOutp = ident, newOcc, newOccMal, getDoor;
SIGNALLIST RoomOccInp = getOcc, newMotDet, newMotDetMal,
newDoorsOpen;

SIGNALLIST DeskOutp = ident, newTaskLight, newEnergy;
SIGNALLIST DeskInp = setEnergyDiffTime, setTaskLight, getTaskLight;

SIGNALLIST LightOutp = ident, newLight, newEnergy;
SIGNALLIST LightInp = setEnergyDiffTime, getLight, setLight, setDimLight, setCIA;

USE DataTypes;
USE Utils;
USE Sensors;
USE Actuators;

Package CP2 3(5)

Office2D

Office1D

CompLab2D

CompLab1D

SIGNALLIST RoomOutp = ident, getSun, newOccMal, newRoomMalfunction, newRoomEnergyConsumption, newIlluminance, newEntrance;

SIGNALLIST RoomInp = newSun, newSunDetMal, setRoomDelay, setRoomReoccDelay, setDefaultLightScene, setDefaultAmbientLightLevel, setRoomEnergyDiffTime, getIlluminance, setRoomLight;

USE DataTypes;
USE Utils;
USE Sensors;
USE Actuators;

Package CP2

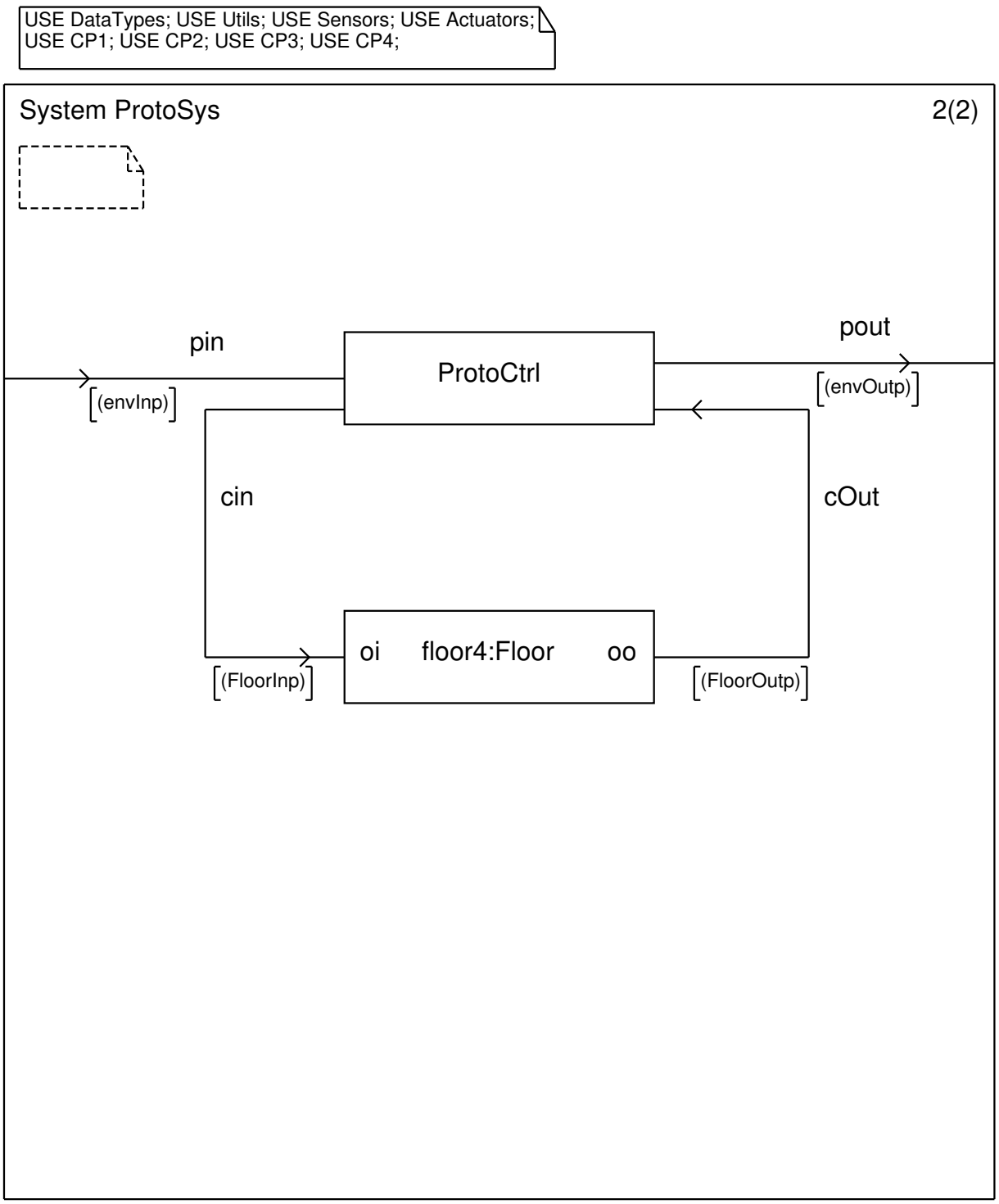
USE DataTypes;
USE Utils;
USE Sensors;
USE Actuators;

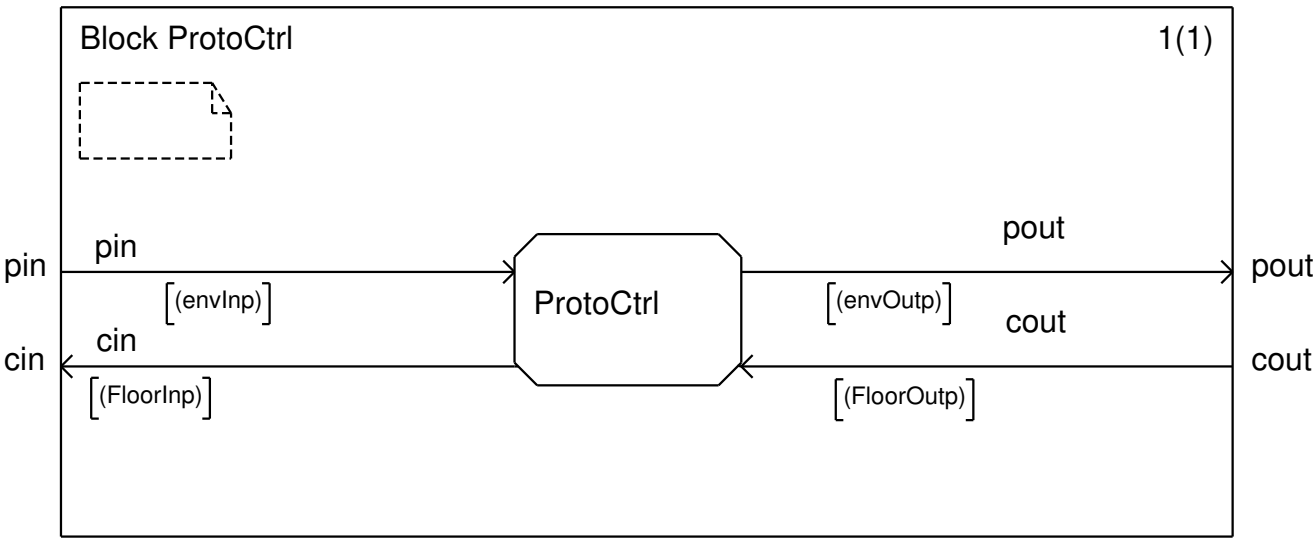
Package CP2 5(5)

```
USE DataTypes; USE Utils; USE Sensors; USE Actuators;
USE CP1; USE CP2; USE CP3; USE CP4;
```

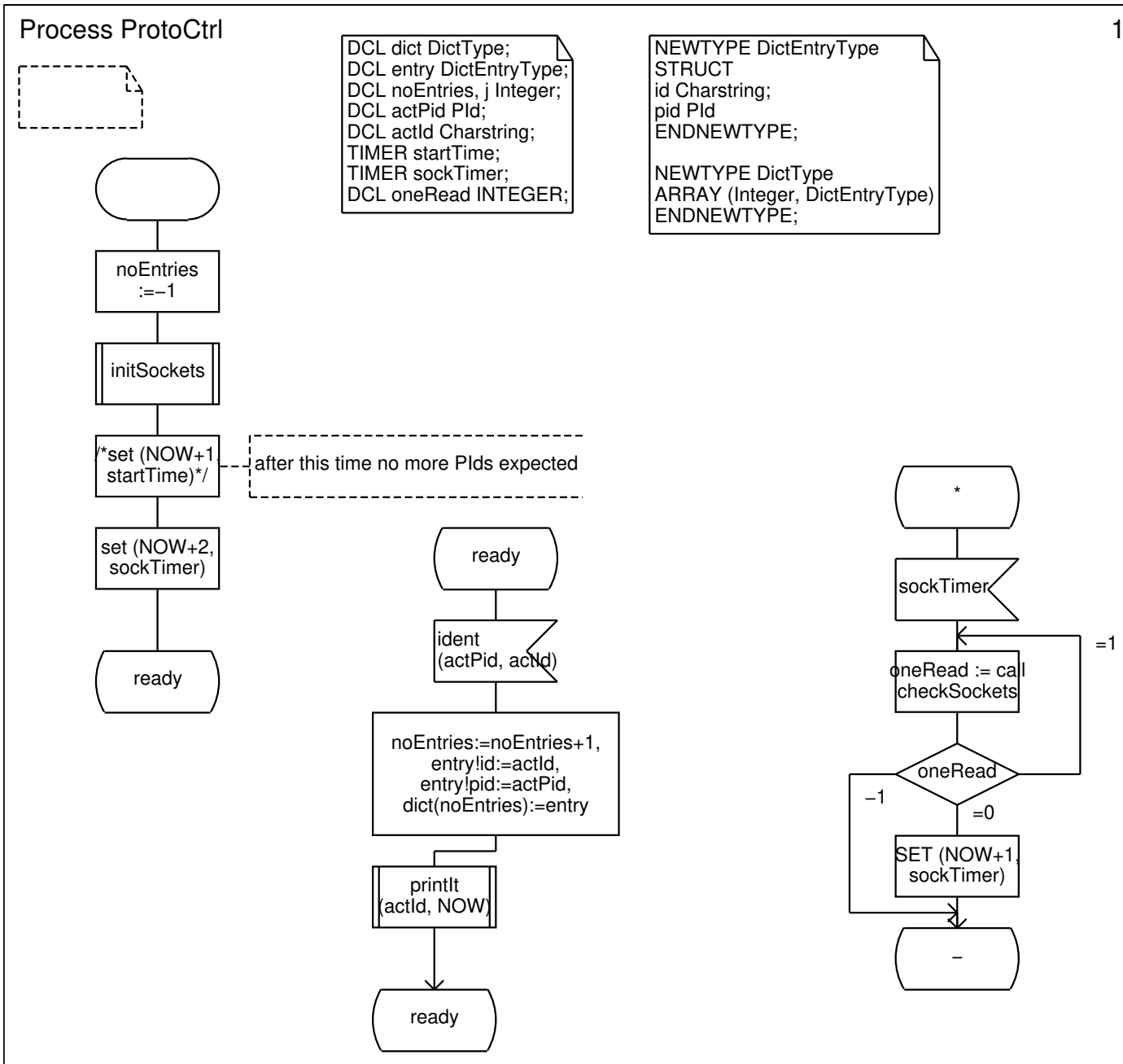
System ProtoSys 1(2)

```
/*ProtoSys signals*/
SIGNAL getPhysVal (CharString);
    /* requests the phys val of the given sensor */
SIGNAL physValIn(Charstring, Analog);
    /* Analog and binary inputs (objectname, value)*/
SIGNAL analogOut(Charstring,Analog), binaryOut(Charstring, Binary);
    /* Binary and analog outputs (objectname, value)*/
SIGNALLIST envInp= physValIn;
SIGNALLIST envOutp= analogOut, binaryOut, getPhysVal;
```

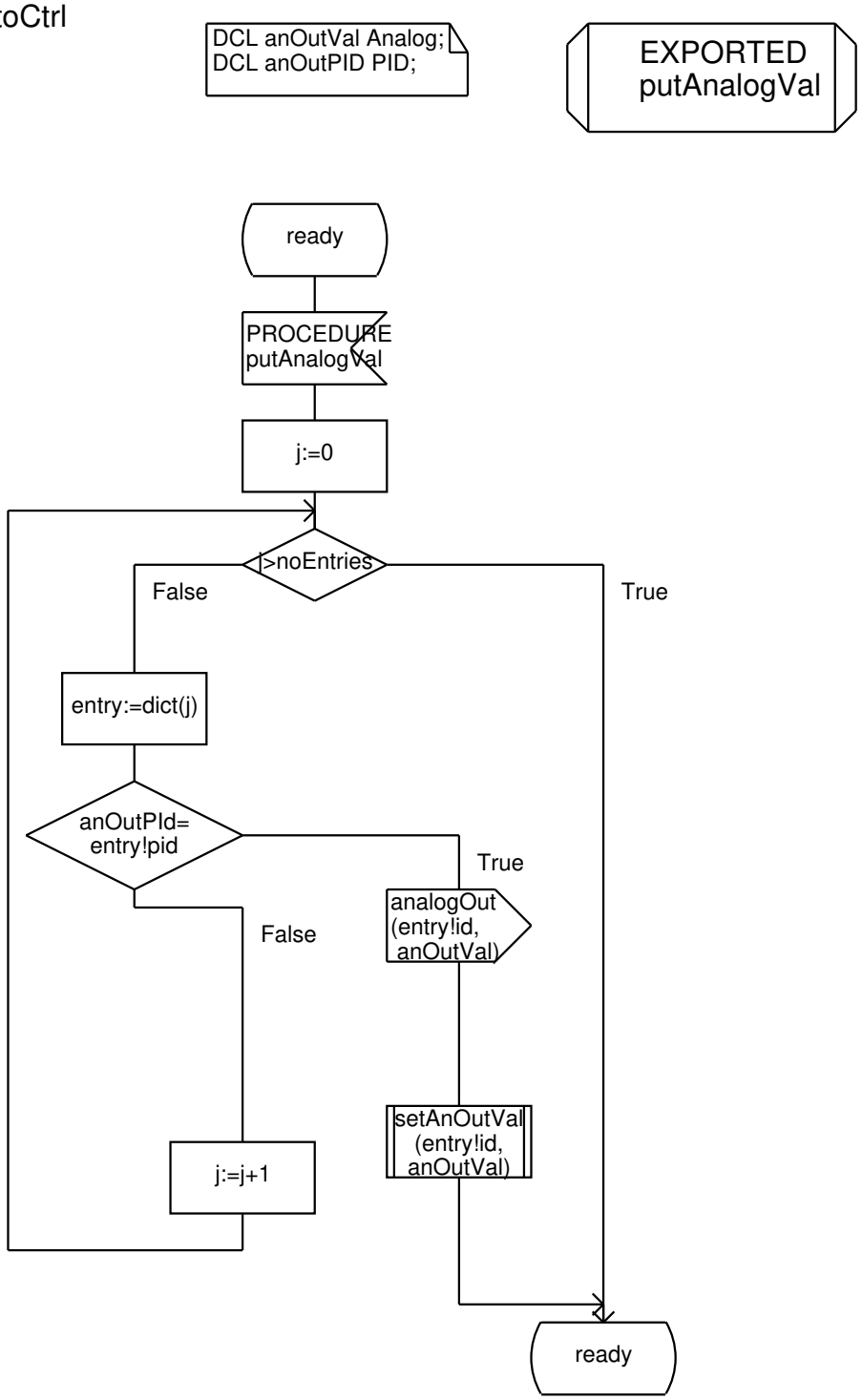

USE DataTypes; USE Utils; USE Sensors; USE Actuators;
USE CP1; USE CP2; USE CP3; USE CP4;
System ProtoSys
2(2)
pin
ProtoCtrl
pout
(envInp)
(envOutp)
cin
cOut
oi
floor4:Floor
oo
(FloorInp)
(FloorOutp)

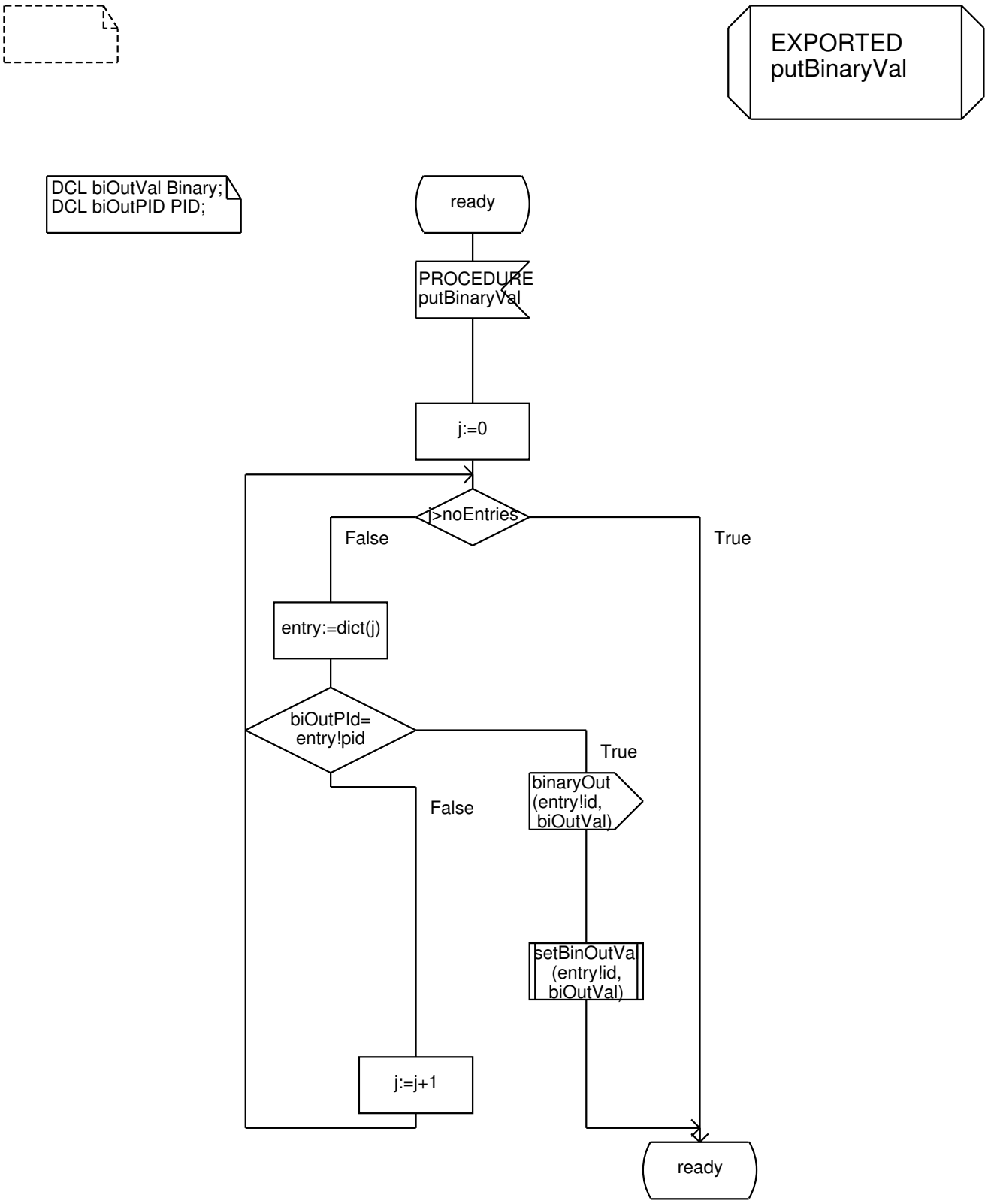

Block ProtoCtrl
1(1)
pin
pin
(envInp)
cin
cin
(FloorInp)
ProtoCtrl
pout
(envOutp)
pout
cout
cout
(FloorOutp)

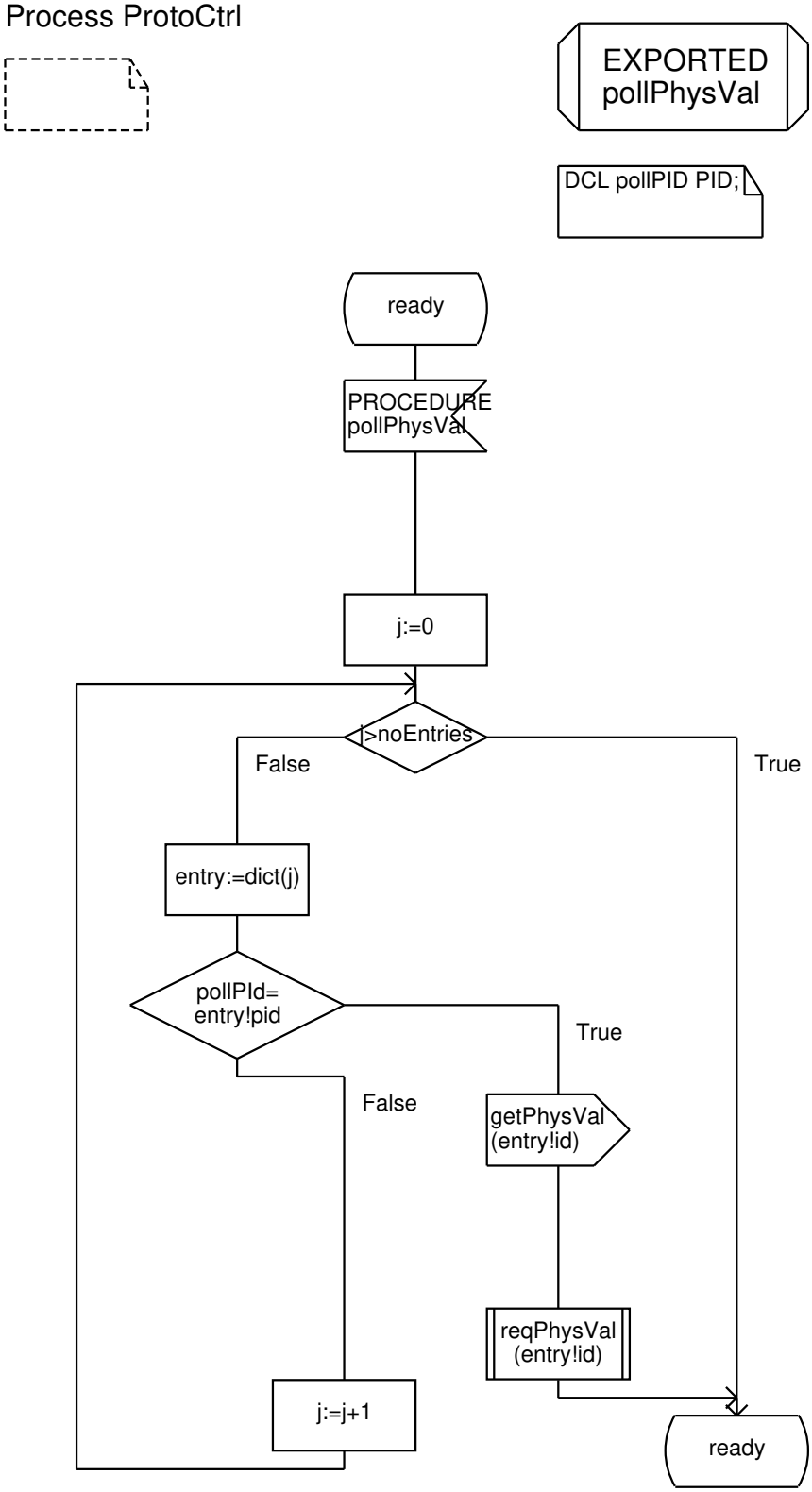
Process ProtoCtrl

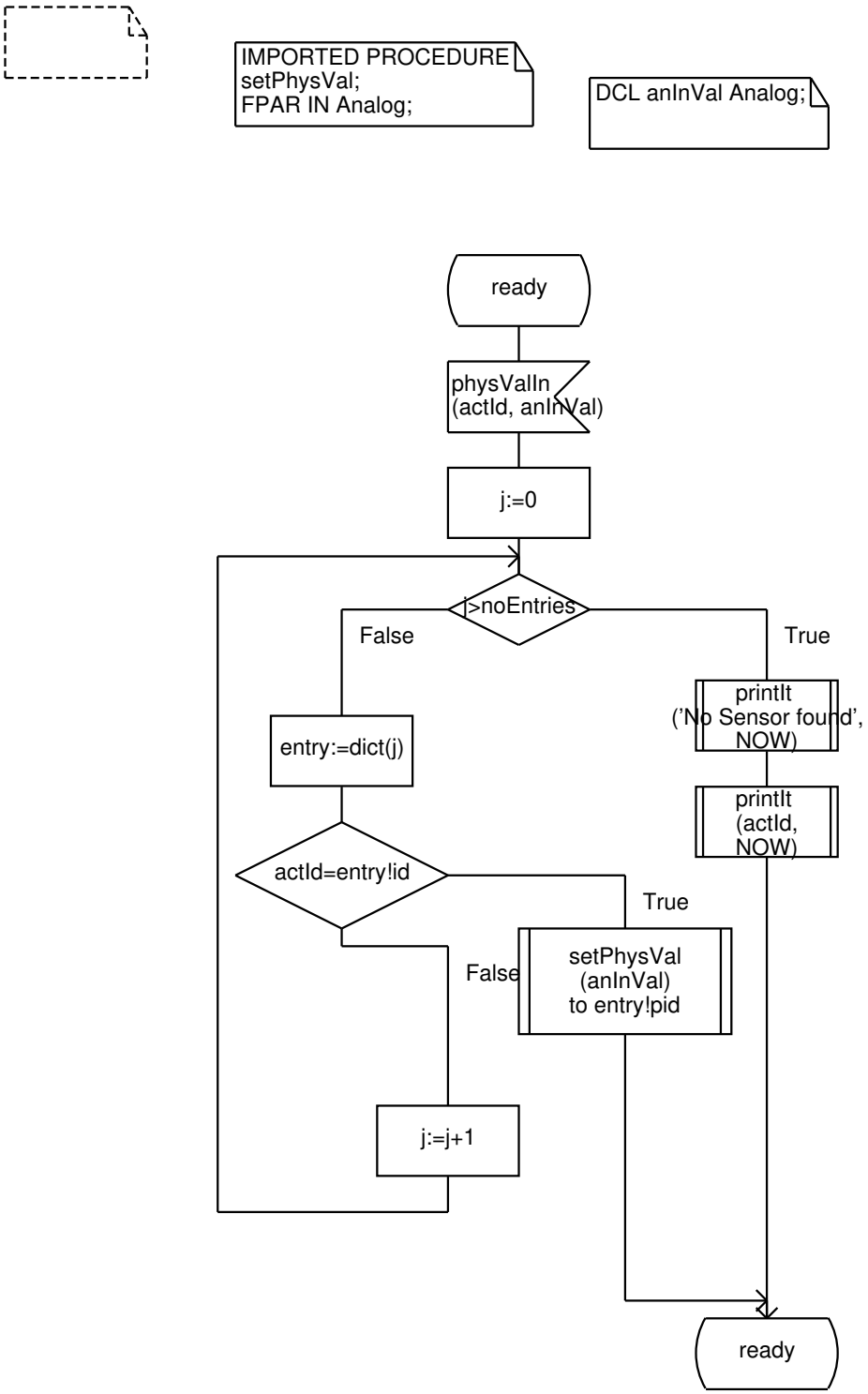


Process ProtoCtrl

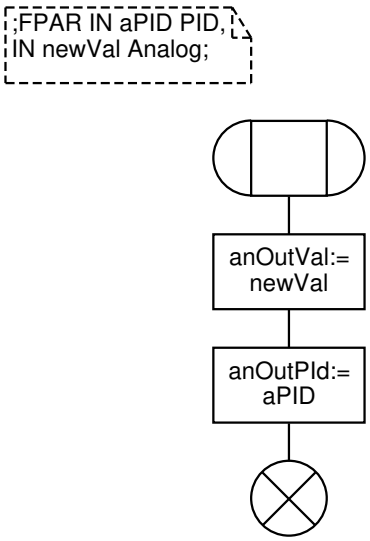



EXPORTED
putBinaryVal
DCL biOutVal Binary;
DCL biOutPID PID;
ready
PROCEDURE
putBinaryVal
j:=0
j>noEntries
False
True
entry:=dict(j)
biOutPId=
entry!pid
True
False
binaryOut
(entry!id,
biOutVal)
setBinOutVal
(entry!id,
biOutVal)
j:=j+1
ready

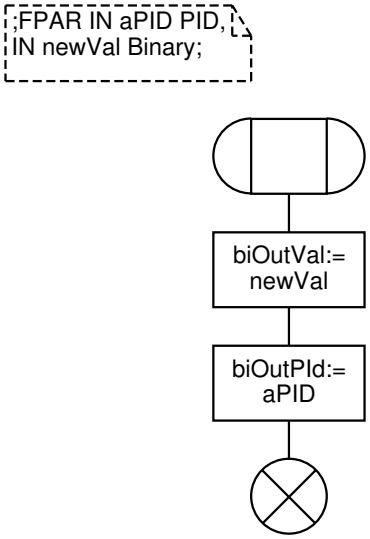
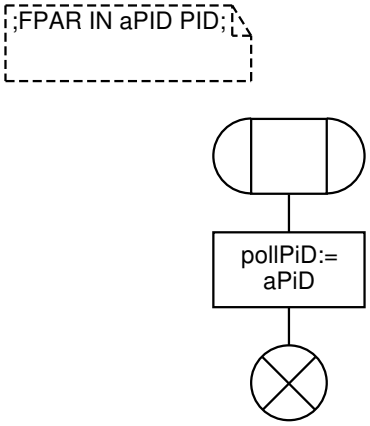

EXPORTED
pollPhysVal
DCL pollPID PID;
ready
PROCEDURE
pollPhysVal
j:=0
j>noEntries
False
True
entry:=dict(j)
pollPId=
entry!pid
True
False
getPhysVal
(entry!id)
reqPhysVal
(entry!id)
j:=j+1
ready


IMPORTED PROCEDURE
setPhysVal;
FPAR IN Analog;
DCL anInVal Analog;
ready
physValIn
(actId, anInVal)
j:=0
j>noEntries
False
True
printIt
('No Sensor found',
NOW)
entry:=dict(j)
printIt
(actId,
NOW)
actId=entry!id
True
False
setPhysVal
(anInVal)
to entry!pid
j:=j+1
ready

Procedure putAnalogVal 1(1)



Procedure putBinaryVal 1(1)



Procedure pollPhysVal 1(1)