Table of Contents

Diagram Structure

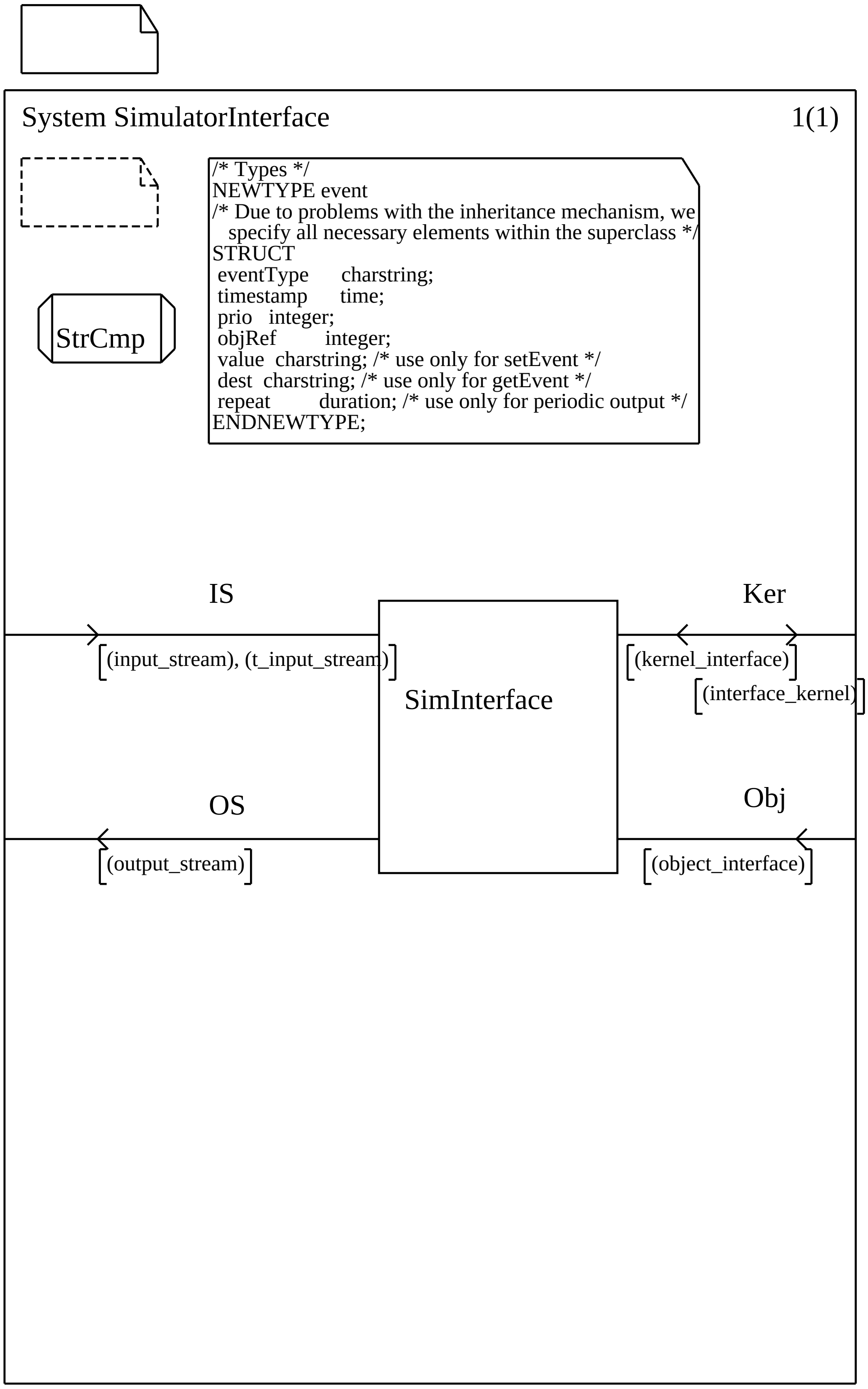
- System SimulatorInterface — simulatorinterface.ssy
  - Procedure StrCmp — strcmp.spd
  - Block SimInterface — siminterface.sbk
    - Block Dictionary — dictionary.sbk
      - Process DictionaryP — dictionaryp.spr
    - Block InputManager — inputmanager.sbk
      - Process Type ReadCST — readcst.spt
      - Process Type ReadFileT — readfilet.spt
        - Procedure WaitForNextRead — waitfornextread.spd
      - Process Type ReadHumanT — readhumant.spt
      - Process Type ReadInputT — readinputt.spt
      - Process TransformInput — transforminput.spr
      - x:y Process Instance ReadCS (0) : ReadCST
      - x:y Process Instance ReadFile (0) : ReadFileT
      - x:y Process Instance ReadHuman (0) : ReadHumanT
    - Block OutputManager — outputmanager1.sbk
      - Process Type Chgd_Value_Handler_T — chgd_value_handler_t.spt
        - Procedure sendToDestinationsOf — sendtodestinationsof.spd
      - Process Type Dic_Handler_T — dic_handler_t.spt
        - Procedure add — add.spd
        - Procedure get_value — get_value.spd
      - Process Type Msg_Handler_T — msg_handler_t.spt
      - Process Type Write_Stream_T — write_stream_t.spt
      - Process Destination_Handler — destination_handler.spr
        - Procedure AddProcess — addprocess.spd
        - Procedure DestExists — destexists.spd
        - Procedure getPId — getpid.spd
      - Process NGets_Handler — ngets_handler.spr
      - x:y Process Instance Chgd_Value_Handler : Chgd_Value_Handler_T
      - x:y Process Instance Msg_Handler : MSG_Handler_T
      - x:y Process Instance Write_Stream (0) : Write_Stream_T

Associated Documents

- Block OutputManager — outputmanager.sbk

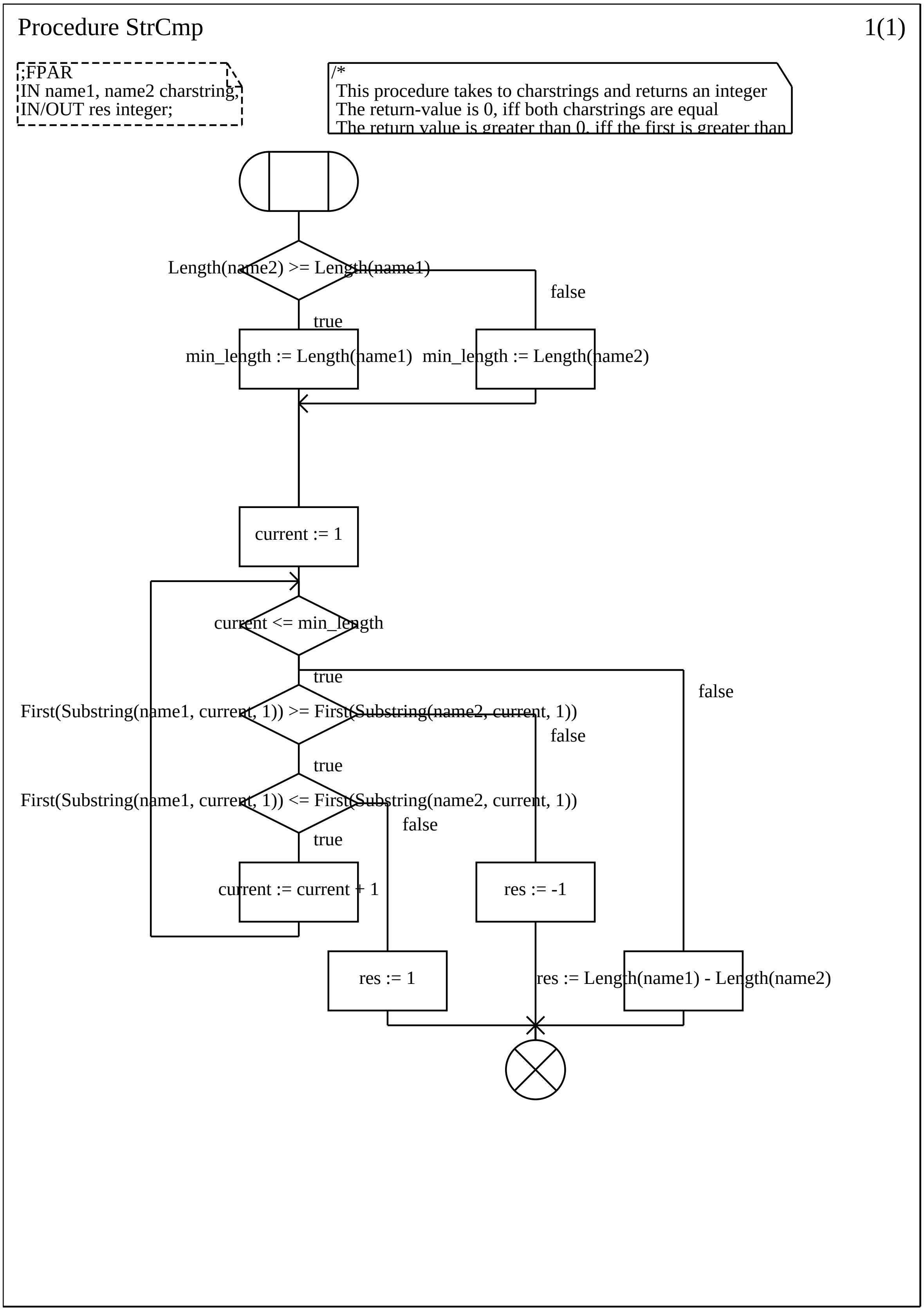
1

MSC Changed_Value	changed_value.msc
MSC Error	error.msc
MSC Get	get.msc
MSC Halt	halt.msc
MSC Init_Successful	init.msc
MSC Log	log.msc
MSC Read_File	read_file.msc
MSC Register	register.msc
MSC Set1	set.msc
MSC Start1	start.msc
MSC Stop1	stop.msc


System SimulatorInterface
1(1)
StrCmp
/* Types */
NEWTYPE event
/* Due to problems with the inheritance mechanism, we
specify all necessary elements within the superclass */
STRUCT
eventType charstring;
timestamp time;
prio integer;
objRef integer;
value charstring; /* use only for setEvent */
dest charstring; /* use only for getEvent */
repeat duration; /* use only for periodic output */
ENDNEWTYPE;
IS
(input_stream), (t_input_stream)
Ker
(kernel_interface)
(interface_kernel)
SimInterface
OS
(output_stream)
Obj
(object_interface)

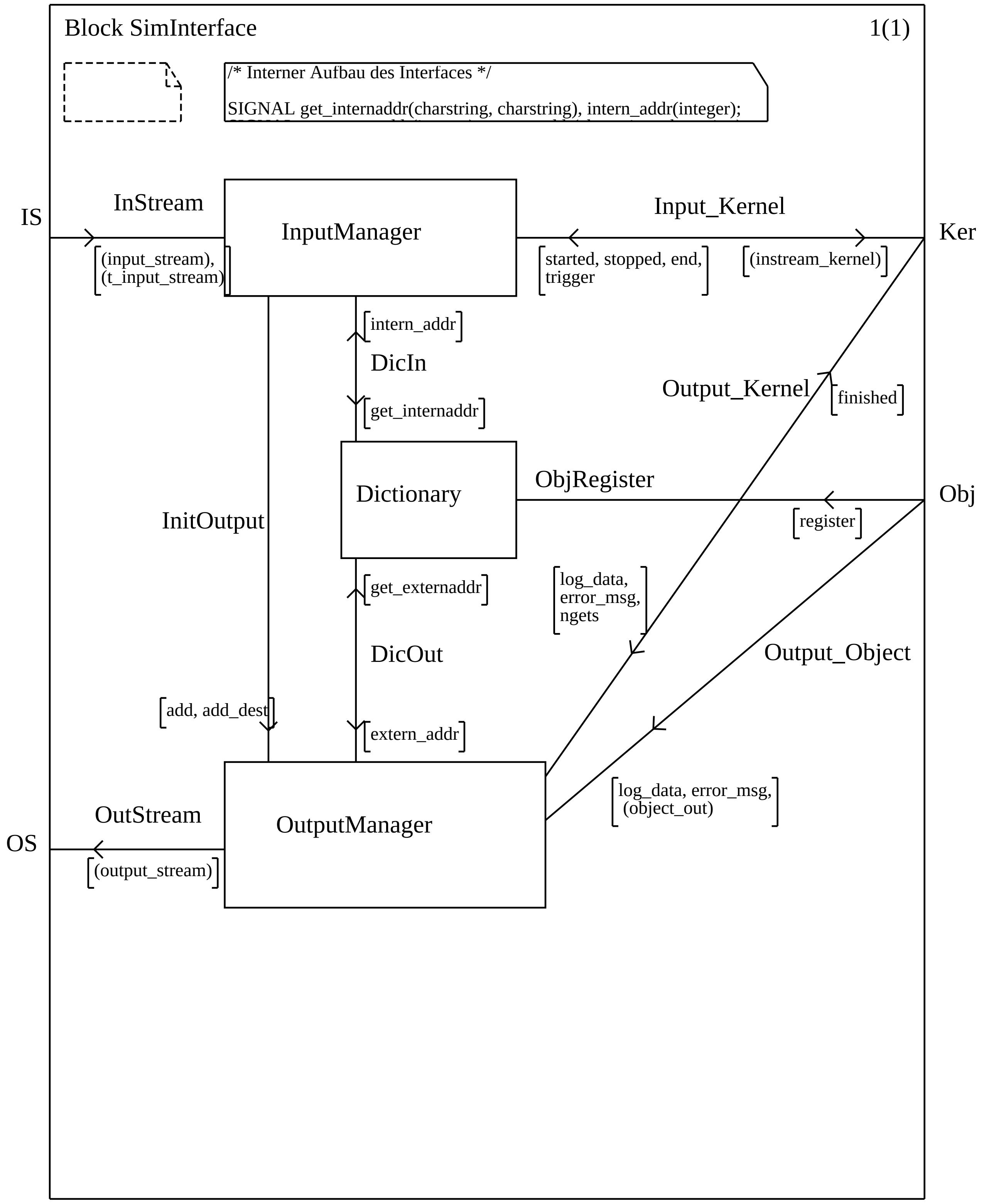
```
/* Types */
NEWTYPE event
/* Due to problems with the inheritance mechanism, we
   specify all necessary elements within the superclass */
STRUCT
 eventType      charstring;
 timestamp      time;
 prio   integer;
 objRef         integer;
 value  charstring; /* use only for setEvent */
 dest  charstring; /* use only for getEvent */
 repeat         duration; /* use only for periodic output */
ENDNEWTYPE;

/*
NEWTYPE getEvent
INHERITS event OPERATORS ALL ADDING
ENDNEWTYPE getEvent;

NEWTYPE setEvent
INHERITS event OPERATORS ALL ADDING
ENDNEWTYPE setEvent;

NEWTYPE stopEvent
INHERITS event OPERATORS ALL ADDING
ENDNEWTYPE stopEvent;

NEWTYPE haltEvent
INHERITS event OPERATORS ALL ADDING
ENDNEWTYPE haltEvent;

NEWTYPE readEvent
INHERITS event OPERATORS ALL ADDING
ENDNEWTYPE readEvent;
*/


/* input-stream signals */
/* untimed */
SIGNAL init,
       init_in_stream(charstring, charstring, integer),
       init_out_stream(charstring, charstring),
       init_periodic(duration, charstring, charstring, charstring),
       init_state_change(charstring, charstring, charstring),
       init_sim_interval(time, time),
       init_slack(time),
       init_timing_factor(real),
       init_logging_flag(boolean),
       endinit;
SIGNAL poll(charstring, charstring, charstring, charstring),
       set_value(charstring, charstring, charstring);
SIGNAL startsig, stopsig, halt;

/*timed */
SIGNAL t_init(time),
       t_init_in_stream(time, charstring, charstring, integer),
       t_init_out_stream(time, charstring, charstring),
       t_init_periodic(time, duration, charstring, charstring, charstring),
       t_init_state_change(time, charstring, charstring, charstring),
       t_init_sim_interval(time, time, time),
       t_init_slack(time, time),
       t_init_timing_factor(time, real),
       t_init_logging_flag(time, boolean),
       t_endinit(time);
SIGNAL t_poll(time, charstring, charstring, charstring, charstring),
```

```
      t_set_value(time, charstring, charstring, charstring);
SIGNAL t_startsig(time), t_stopsig(time), t_halt(time);


/* output-stream signals */
SIGNAL output_value(time, charstring, charstring, charstring),
      result(charstring);
SIGNAL log(charstring), ack, error_message(charstring);

/* kernel-interface signals */
SIGNAL trigger(event), ngets(integer);
SIGNAL started, stopped, end;
SIGNAL ok;

/* interface-kernel signals */
SIGNAL ik_init,
      ik_init_sim_interval(time, time),
      ik_init_slack(time),
      ik_init_timing_factor(real),
      ik_init_logging_flag(boolean),
      ik_endinit;
SIGNAL schedule(event, integer), finished;
SIGNAL ik_start;

/* object-interface signals */
SIGNAL register(charstring, charstring, integer);
SIGNAL obj_value(time, integer, charstring, charstring);
SIGNAL obj_changed_value(time, integer, charstring);

/* general */
SIGNAL log_data(charstring), error_msg(charstring);


SIGNALLIST input_stream =
  init, init_in_stream, init_out_stream, init_periodic,
  init_state_change, init_sim_interval, init_slack,
  init_timing_factor, init_logging_flag, endinit,
  poll, set_value, startsig, stopsig, halt;
SIGNALLIST t_input_stream =
  t_init, t_init_in_stream, t_init_out_stream, t_init_periodic,
  t_init_state_change, t_init_sim_interval, t_init_slack,
  t_init_timing_factor, t_init_logging_flag, t_endinit,
  t_poll, t_set_value, t_startsig, t_stopsig, t_halt;
SIGNALLIST output_stream =
  output_value, result, log, ack, error_message;
SIGNALLIST kernel_interface =
  trigger, ngets, started, stopped, end, log_data, ok, error_msg;
SIGNALLIST interface_kernel =
  ik_init, ik_init_sim_interval, ik_init_slack,
  ik_init_timing_factor, ik_init_logging_flag, ik_endinit,
  schedule, finished, ik_start;
SIGNALLIST object_interface =
  register, obj_value, obj_changed_value, log_data, error_msg;
```

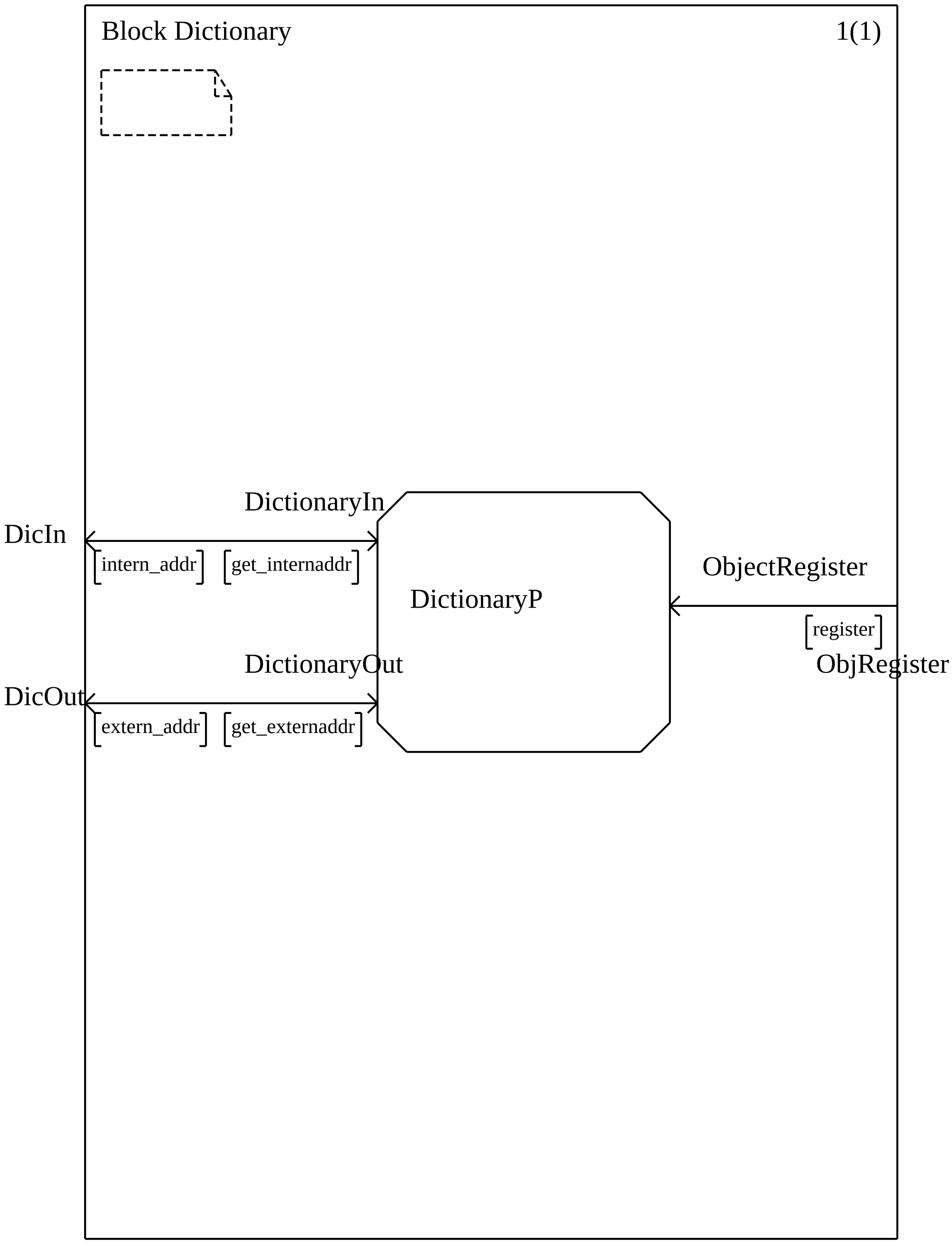


Procedure StrCmp
1(1)
;FPAR
IN name1, name2 charstring;
IN/OUT res integer;
/*
This procedure takes to charstrings and returns an integer
The return-value is 0, iff both charstrings are equal
The return value is greater than 0, iff the first is greater than
Length(name2) >= Length(name1)
false
true
min_length := Length(name1)
min_length := Length(name2)
current := 1
current <= min_length
true
false
First(Substring(name1, current, 1)) >= First(Substring(name2, current, 1))
false
true
First(Substring(name1, current, 1)) <= First(Substring(name2, current, 1))
false
true
current := current + 1
res := -1
res := 1
res := Length(name1) - Length(name2)

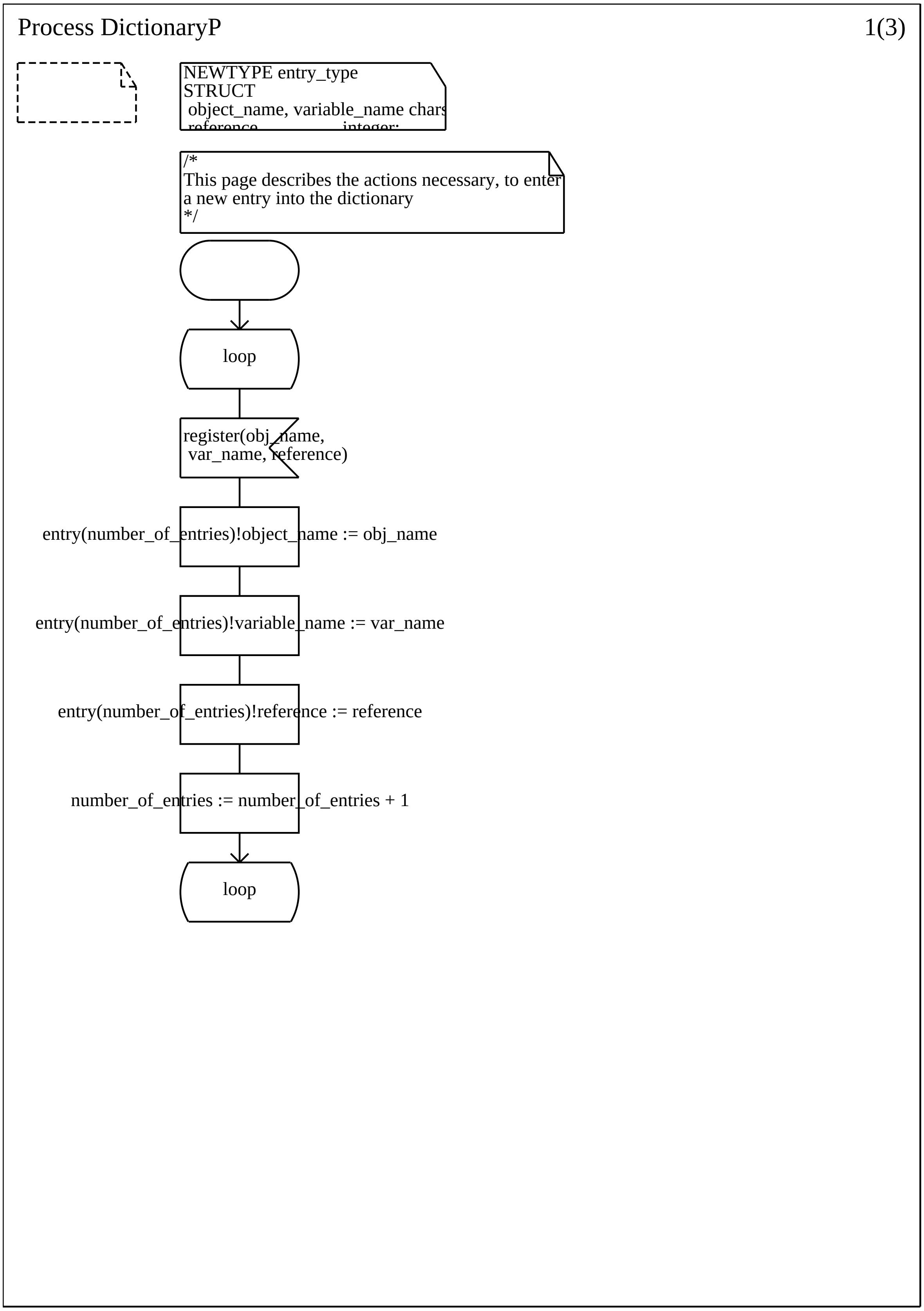
```
/*
 This procedure takes to charstrings and returns an integer
 The return-value is 0, iff both charstrings are equal
 The return value is greater than 0, iff the first is greater than the second
 The return value is less than 0, iff the first is less than the second
*/

DCL min_length integer; /* minimal length of name1 and name2 */
DCL current integer;    /* current position within the names */
```


Block SimInterface
1(1)
/* Interner Aufbau des Interfaces */
SIGNAL get_internaddr(charstring, charstring), intern_addr(integer);
IS
InStream
(input_stream),
(t_input_stream)
InputManager
Input_Kernel
started, stopped, end,
trigger
(instream_kernel)
Ker
intern_addr
DicIn
get_internaddr
Output_Kernel
finished
Dictionary
ObjRegister
register
Obj
InitOutput
get_externaddr
log_data,
error_msg,
ngets
DicOut
Output_Object
add, add_dest
extern_addr
log_data, error_msg,
(object_out)
OutStream
OS
OutputManager
(output_stream)

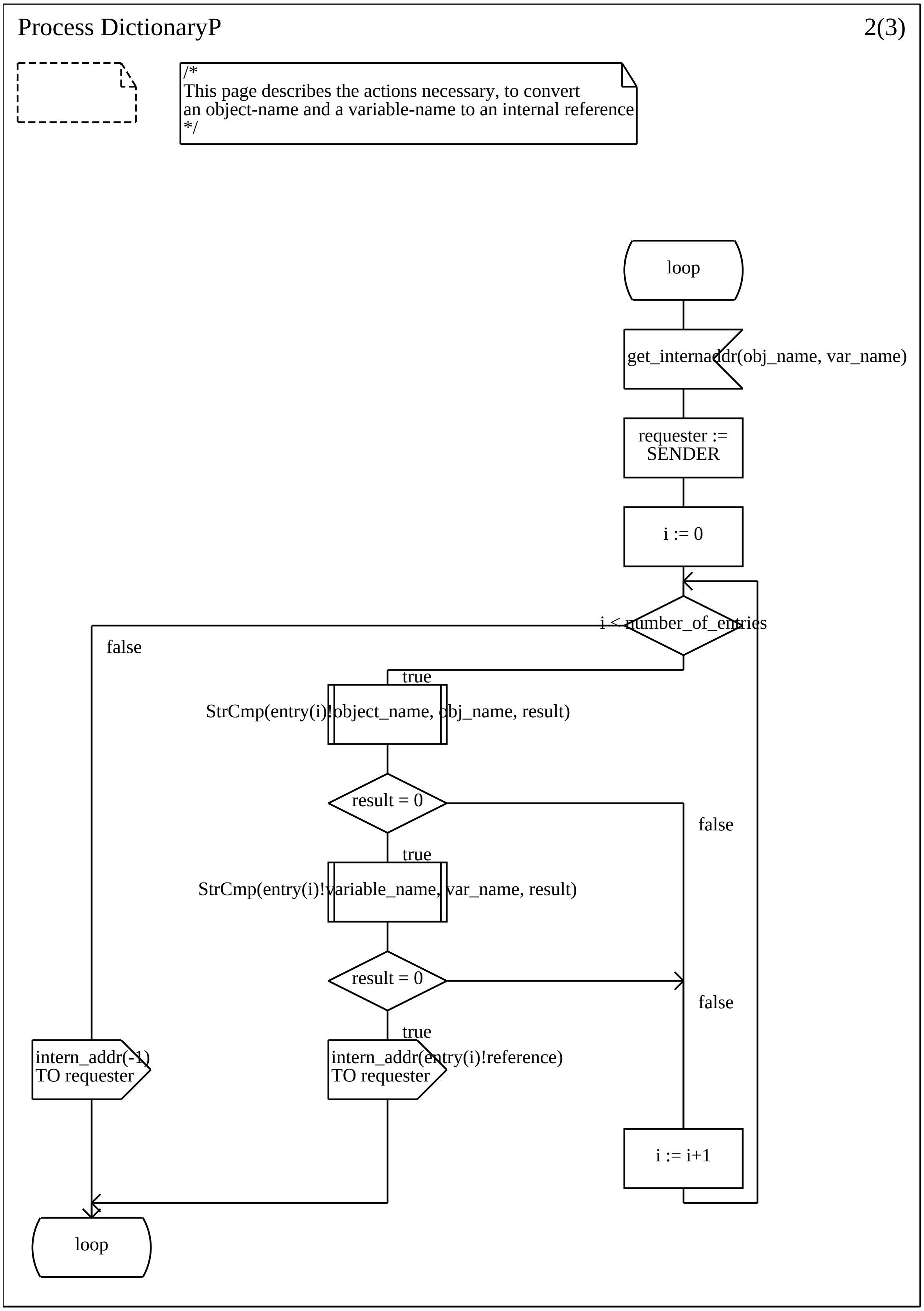
```
/* Interner Aufbau des Interfaces */

SIGNAL get_internaddr(charstring, charstring), intern_addr(integer);
SIGNAL get_externaddr(integer), extern_addr(charstring, charstring);
SIGNAL trigger;

SIGNAL add_dest(charstring); /* Destination */
SIGNAL add(charstring, charstring, charstring); /* ObjectName, VarName, Destination */

SIGNALLIST object_out = obj_value, obj_changed_value;
SIGNALLIST instream_kernel =
  ik_init, ik_init_sim_interval, ik_init_slack,
  ik_init_timing_factor, ik_init_logging_flag, ik_endinit,
  ik_start, schedule;

/* The following signals aren't yet considered:
kernel -> interface: ok kann als acknowledge an die Ausgabe weitergereicht werden
    (ueber Kanal Ker_Msg)
    oder fuer Zustandswechsel innerhalb von ConvertInStream verwendet werden
    (idle -> stopped)
*/
```



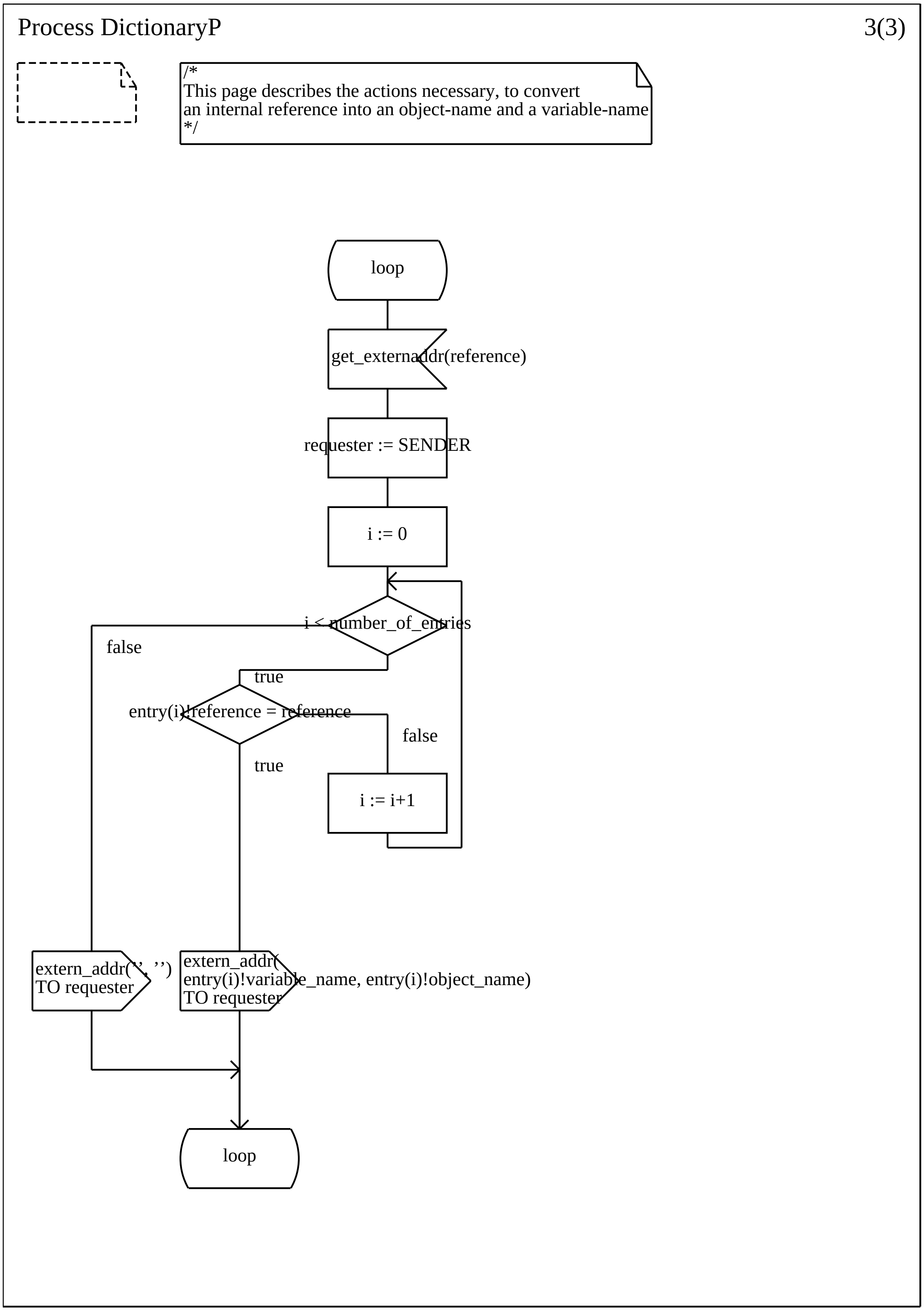
Block Dictionary
1(1)
DictionaryIn
DicIn
intern_addr
get_internaddr
DictionaryP
ObjectRegister
register
ObjRegister
DictionaryOut
DicOut
extern_addr
get_externaddr

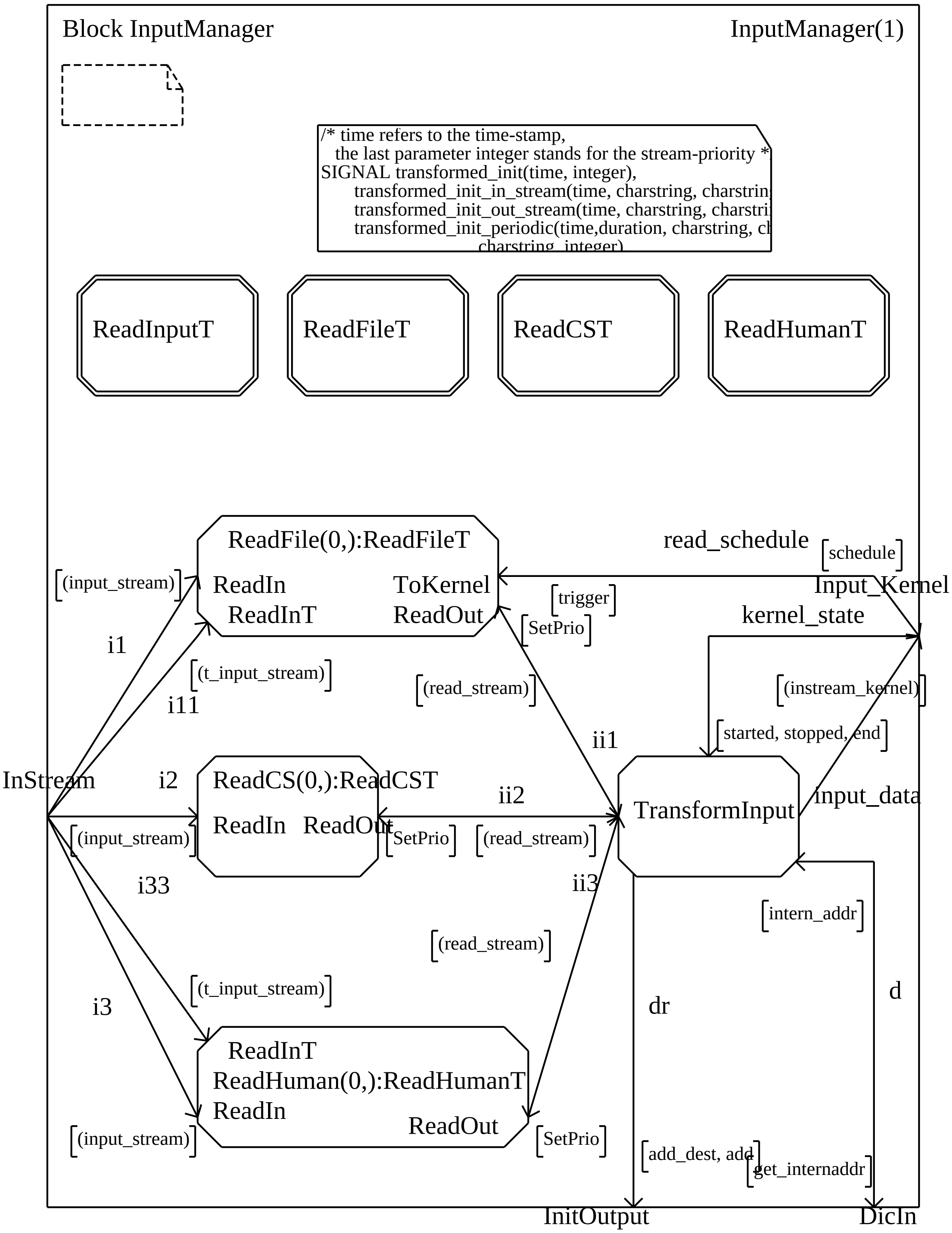

Process DictionaryP
1(3)
NEWTYPE entry_type
STRUCT
object_name, variable_name chars
/*
This page describes the actions necessary, to enter
a new entry into the dictionary
*/
loop
register(obj_name, var_name, reference)
entry(number_of_entries)!object_name := obj_name
entry(number_of_entries)!variable_name := var_name
entry(number_of_entries)!reference := reference
number_of_entries := number_of_entries + 1
loop

```
NEWTYPE entry_type
STRUCT
 object_name, variable_name charstring;
 reference                  integer;
ENDNEWTYPE;

NEWTYPE dict
Array(integer, entry_type);
ENDNEWTYPE;

DCL obj_name  charstring;
DCL var_name  charstring;
DCL reference integer;

DCL entry            dict;
DCL number_of_entries integer := 0;

DCL result integer;

DCL i integer; /* Laufvariable */

DCL requester PId; /* PId des Anfrageobjektes */
```



Process DictionaryP
2(3)
/*
This page describes the actions necessary, to convert
an object-name and a variable-name to an internal reference
*/
loop
get_internaddr(obj_name, var_name)
requester :=
SENDER
i := 0
i < number_of_entries
false
true
StrCmp(entry(i)!object_name, obj_name, result)
result = 0
false
true
StrCmp(entry(i)!variable_name, var_name, result)
result = 0
false
true
intern_addr(-1)
TO requester
intern_addr(entry(i)!reference)
TO requester
i := i+1
loop


Process DictionaryP
3(3)
/*
This page describes the actions necessary, to convert
an internal reference into an object-name and a variable-name
*/
loop
get_externaddr(reference)
requester := SENDER
i := 0
i < number_of_entries
false
true
entry(i)!reference = reference
false
true
i := i+1
extern_addr('', '')
TO requester
extern_addr(
entry(i)!variable_name, entry(i)!object_name)
TO requester
loop

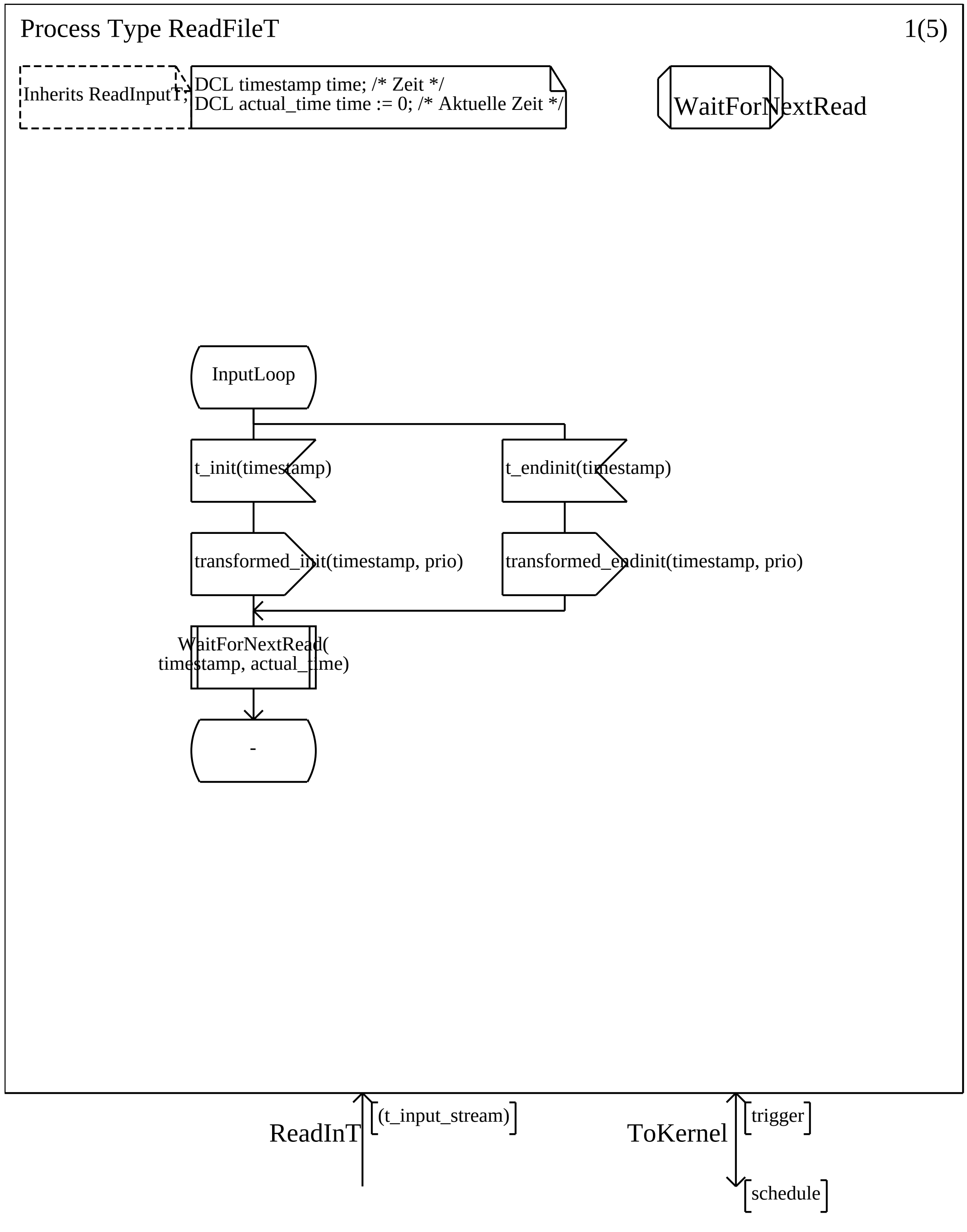


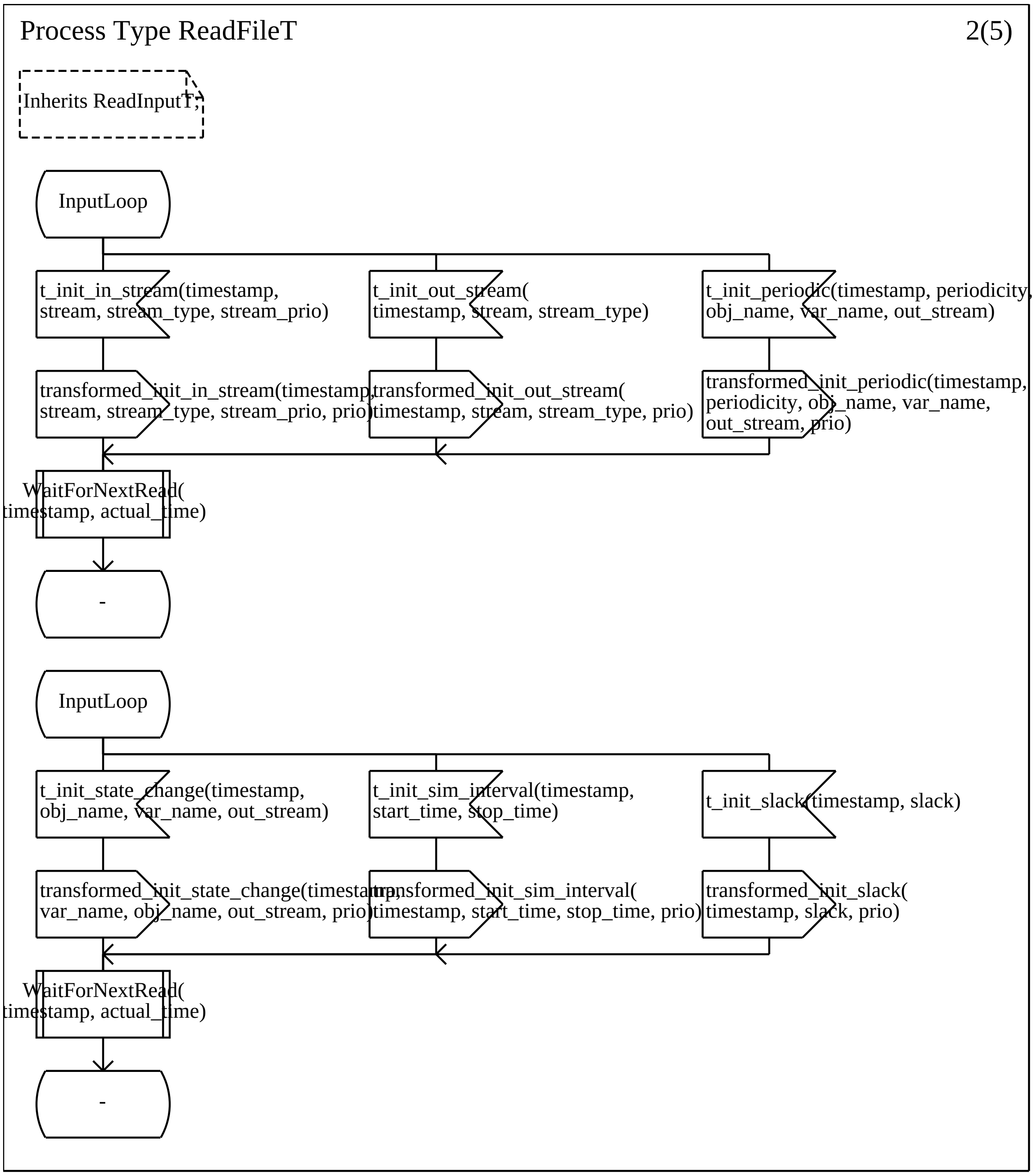
Block InputManager
InputManager(1)
/* time refers to the time-stamp,
the last parameter integer stands for the stream-priority *
SIGNAL transformed_init(time, integer),
transformed_init_in_stream(time, charstring, charstrin
transformed_init_out_stream(time, charstring, charstri
transformed_init_periodic(time,duration, charstring, c
charstring, integer)
ReadInputT
ReadFileT
ReadCST
ReadHumanT
ReadFile(0,):ReadFileT
ReadIn
ReadInT
ToKernel
ReadOut
read_schedule
schedule
Input_Kernel
trigger
SetPrio
kernel_state
(input_stream)
i1
(t_input_stream)
i11
(read_stream)
(instream_kernel)
started, stopped, end
ii1
InStream
i2
ReadCS(0,):ReadCST
ReadIn
ReadOut
ii2
TransformInput
input_data
(input_stream)
SetPrio
(read_stream)
i33
ii3
intern_addr
(read_stream)
(t_input_stream)
i3
dr
d
ReadInT
ReadHuman(0,):ReadHumanT
ReadIn
ReadOut
(input_stream)
SetPrio
add_dest, add
get_internaddr
InitOutput
DicIn

```
/* time refers to the time-stamp,
   the last parameter integer stands for the stream-priority */
SIGNAL transformed_init(time, integer),
       transformed_init_in_stream(time, charstring, charstring, integer, integer),
       transformed_init_out_stream(time, charstring, charstring, integer),
       transformed_init_periodic(time,duration, charstring, charstring,
                              charstring, integer),
       transformed_init_state_change(time, charstring, charstring,
                                charstring, integer),
       transformed_init_sim_interval(time, time, time, integer),
       transformed_init_slack(time, time, integer),
       transformed_init_timing_factor(time, real, integer),
       transformed_init_logging_flag(time, boolean, integer),
       transformed_endinit(time, integer);
SIGNAL transformed_poll(time, charstring, charstring,
                      charstring, charstring, integer),
       transformed_set_value(time, charstring, charstring,
                           charstring, integer);
SIGNAL transformed_startsig(time, integer),
       transformed_stopsig(time, integer),
       transformed_halt(time, integer);

SIGNALLIST read_stream =
 transformed_init, transformed_init_in_stream,
 transformed_init_out_stream, transformed_init_periodic,
 transformed_init_state_change, transformed_init_sim_interval,
 transformed_init_slack, transformed_init_timing_factor,
 transformed_init_logging_flag, transformed_endinit,
 transformed_poll, transformed_set_value,
 transformed_startsig, transformed_stopsig, transformed_halt;

SIGNAL SetPrio(integer);
```

Process Type ReadCST

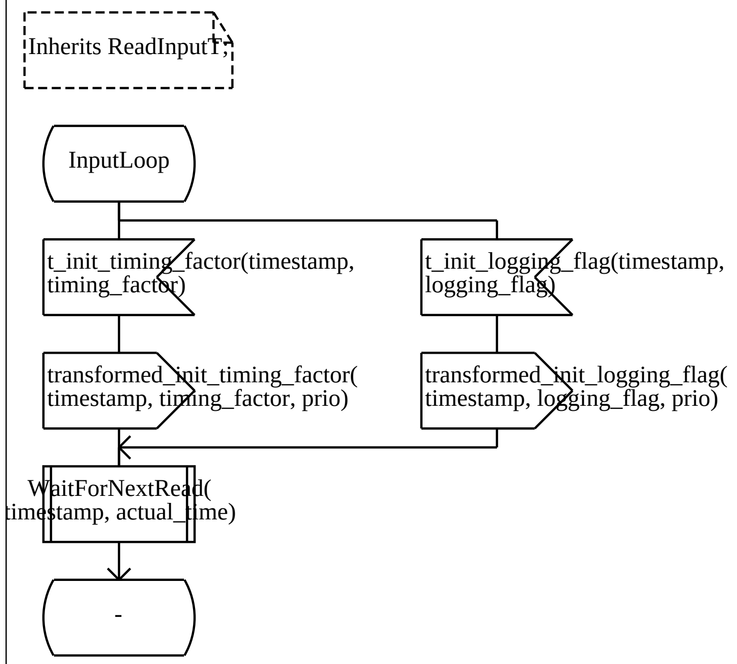
Inherits ReadInputT;


Process Type ReadFileT
1(5)
Inherits ReadInputT;
DCL timestamp time; /* Zeit */
DCL actual_time time := 0; /* Aktuelle Zeit */
WaitForNextRead
InputLoop
t_init(timestamp)
t_endinit(timestamp)
transformed_init(timestamp, prio)
transformed_endinit(timestamp, prio)
WaitForNextRead(
timestamp, actual_time)
-
ReadInT
(t_input_stream)
ToKernel
trigger
schedule

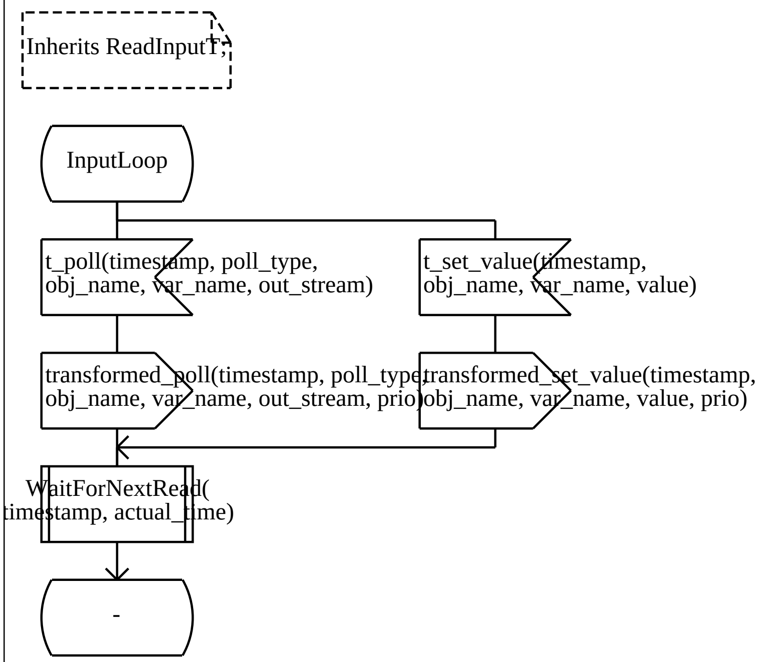


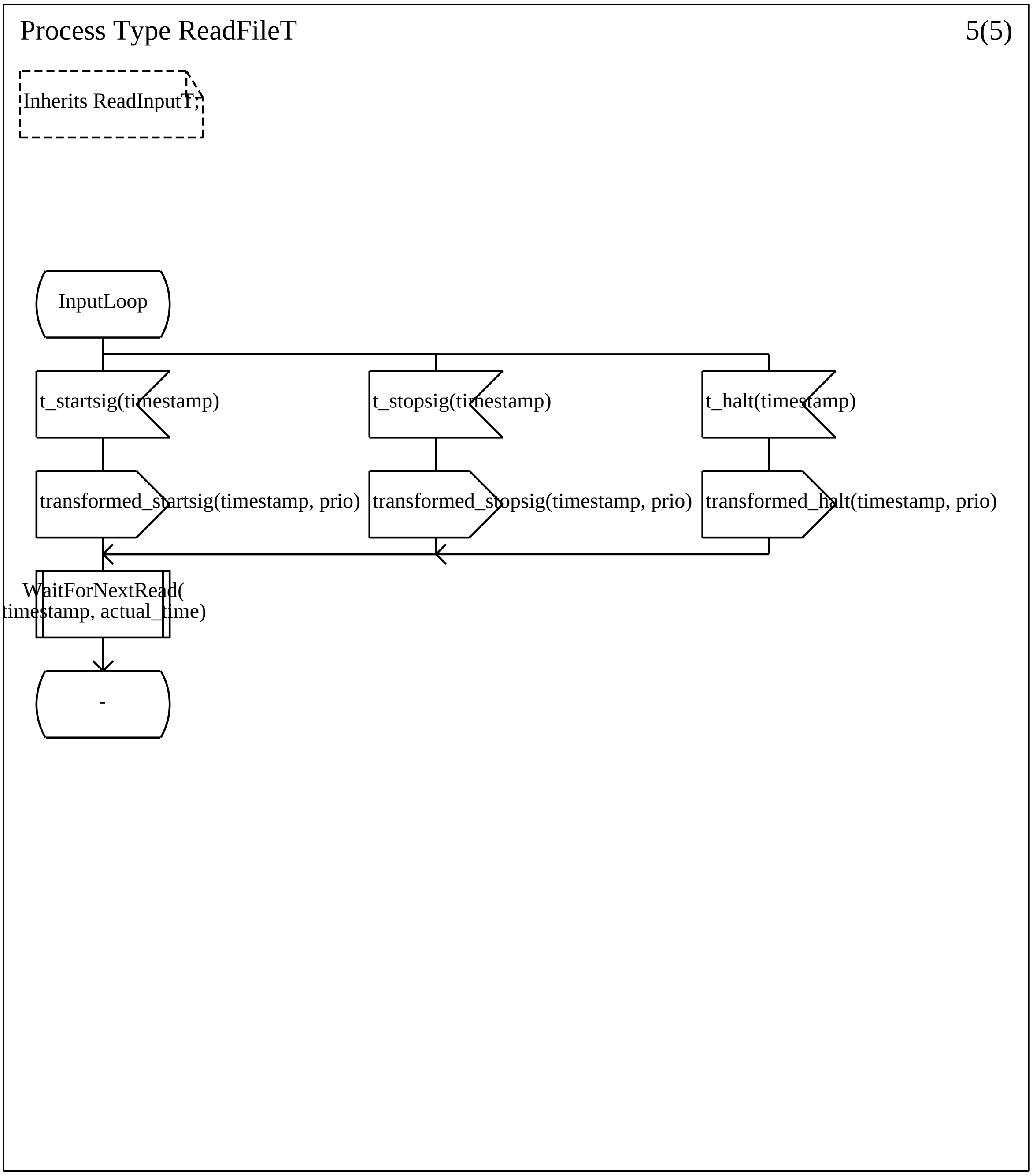
Process Type ReadFileT
2(5)
Inherits ReadInputT;
InputLoop
t_init_in_stream(timestamp, stream, stream_type, stream_prio)
t_init_out_stream(timestamp, stream, stream_type)
t_init_periodic(timestamp, periodicity, obj_name, var_name, out_stream)
transformed_init_in_stream(timestamp, stream, stream_type, stream_prio, prio)
transformed_init_out_stream(timestamp, stream, stream_type, prio)
transformed_init_periodic(timestamp, periodicity, obj_name, var_name, out_stream, prio)
WaitForNextRead(timestamp, actual_time)
-
InputLoop
t_init_state_change(timestamp, obj_name, var_name, out_stream)
t_init_sim_interval(timestamp, start_time, stop_time)
t_init_slack(timestamp, slack)
transformed_init_state_change(timestamp, var_name, obj_name, out_stream, prio)
transformed_init_sim_interval(timestamp, start_time, stop_time, prio)
transformed_init_slack(timestamp, slack, prio)
WaitForNextRead(timestamp, actual_time)
-

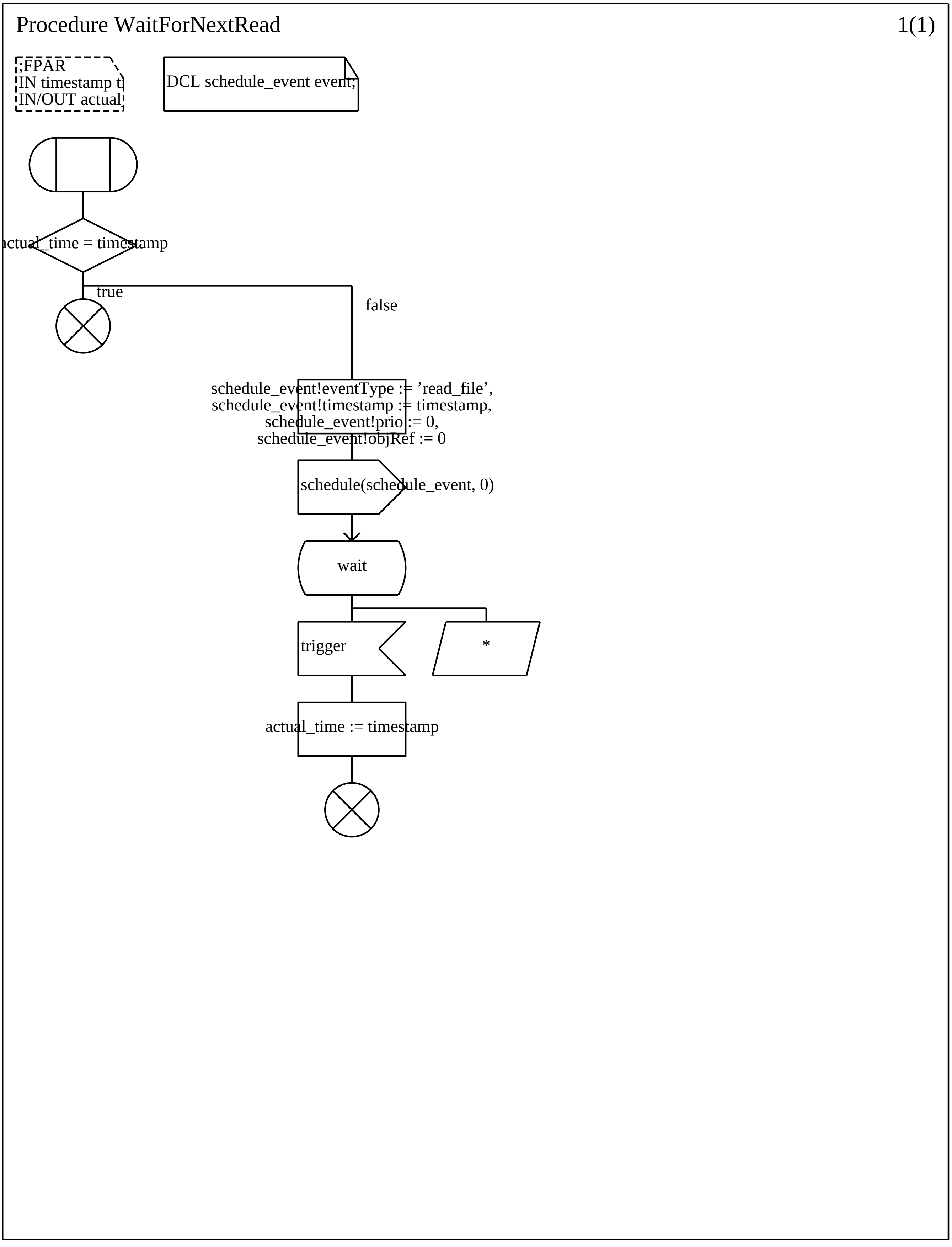
Process Type ReadFileT 3(5)



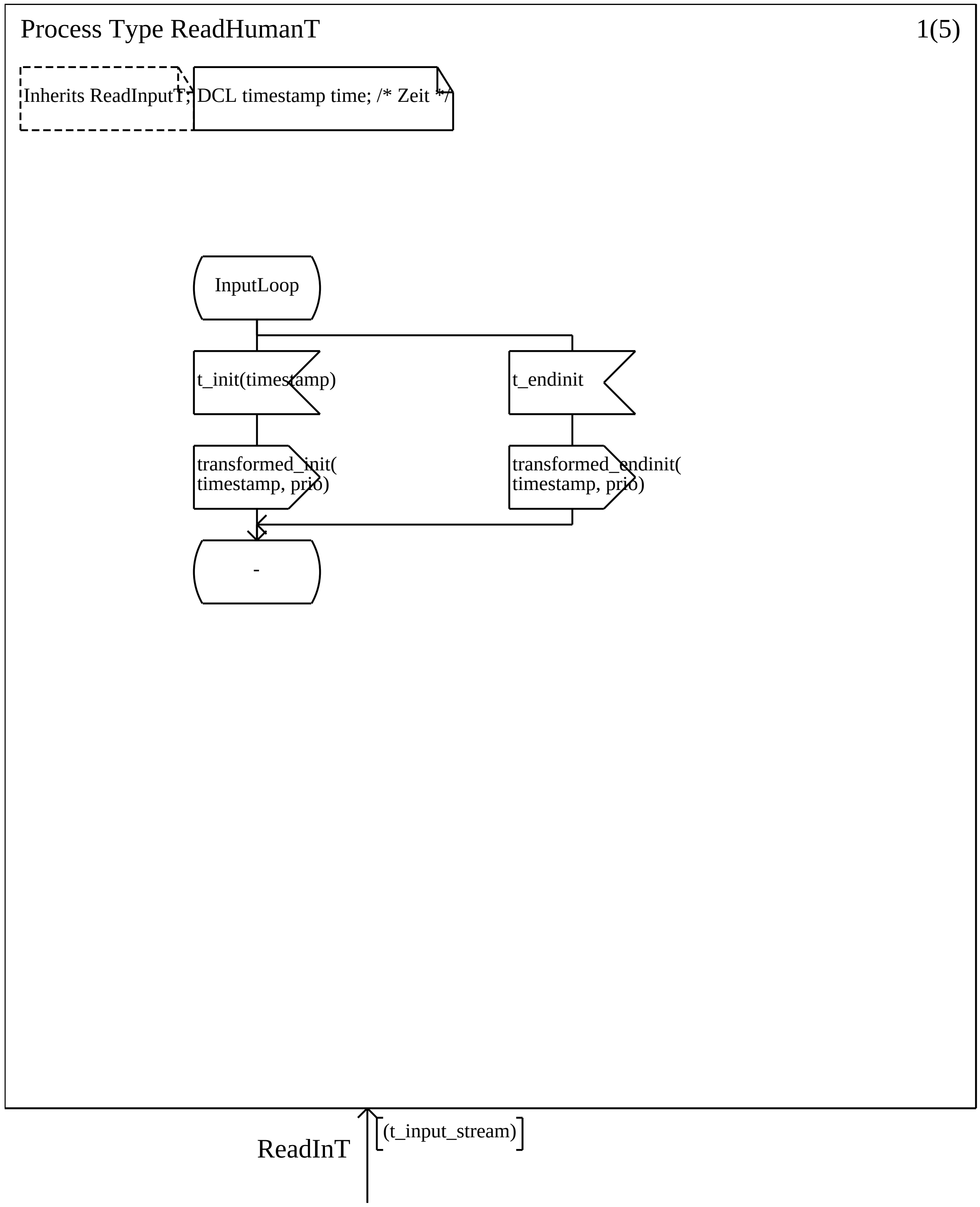
Process Type ReadFileT 4(5)

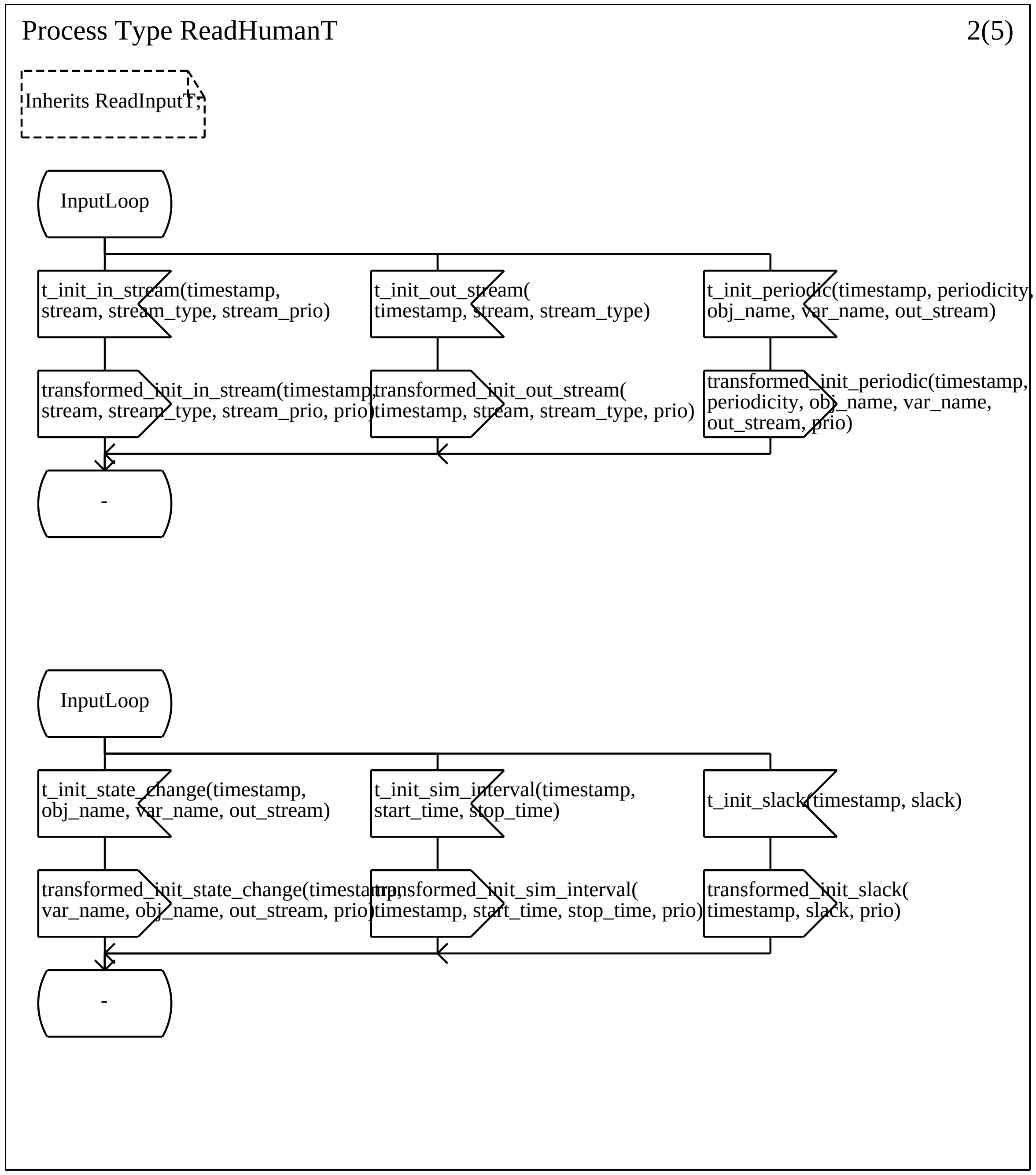



Process Type ReadFileT
5(5)
Inherits ReadInputT;
InputLoop
t_startsig(timestamp)
t_stopsig(timestamp)
t_halt(timestamp)
transformed_startsig(timestamp, prio)
transformed_stopsig(timestamp, prio)
transformed_halt(timestamp, prio)
WaitForNextRead(
timestamp, actual_time)
-

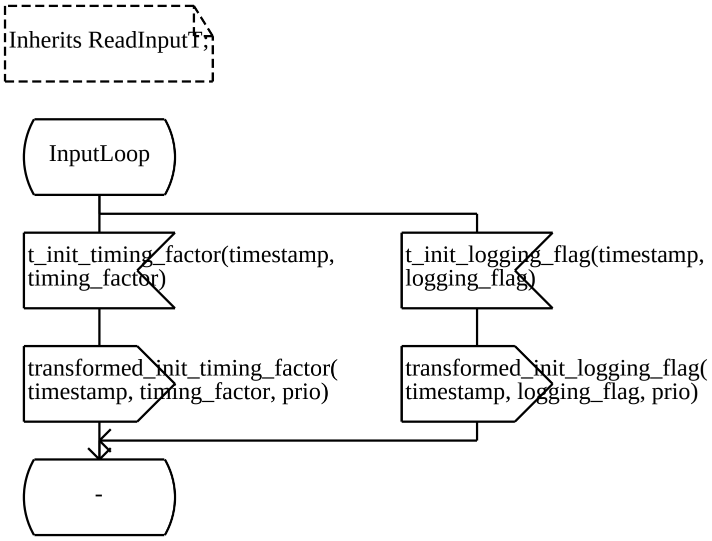

Procedure WaitForNextRead
1(1)
;FPAR
IN timestamp t
IN/OUT actual
DCL schedule_event event;
actual_time = timestamp
true
false
schedule_event!eventType := 'read_file',
schedule_event!timestamp := timestamp,
schedule_event!prio := 0,
schedule_event!objRef := 0
schedule(schedule_event, 0)
wait
trigger
*
actual_time := timestamp

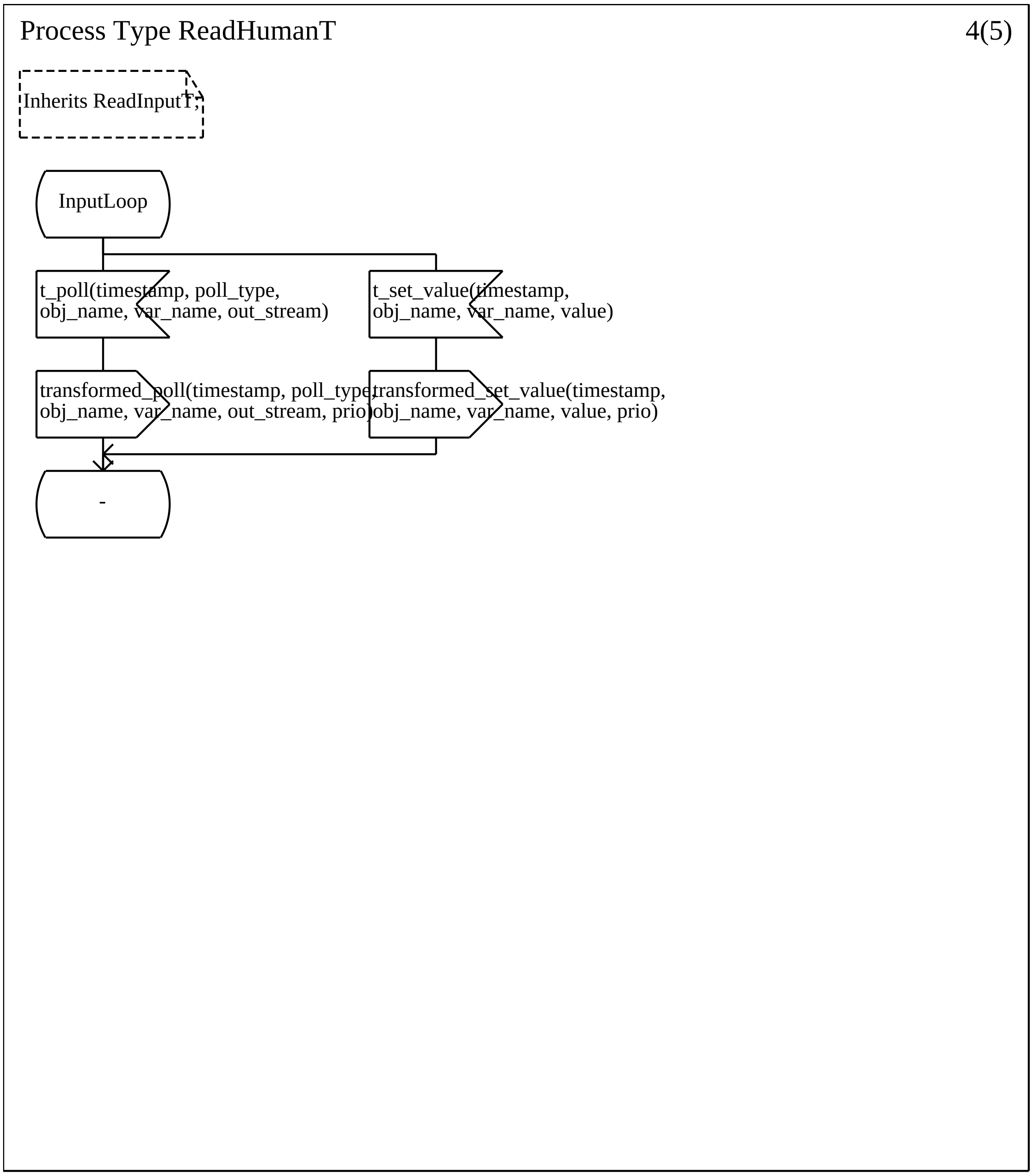
```
;FPAR
IN timestamp time,
IN/OUT actual_time time;
```

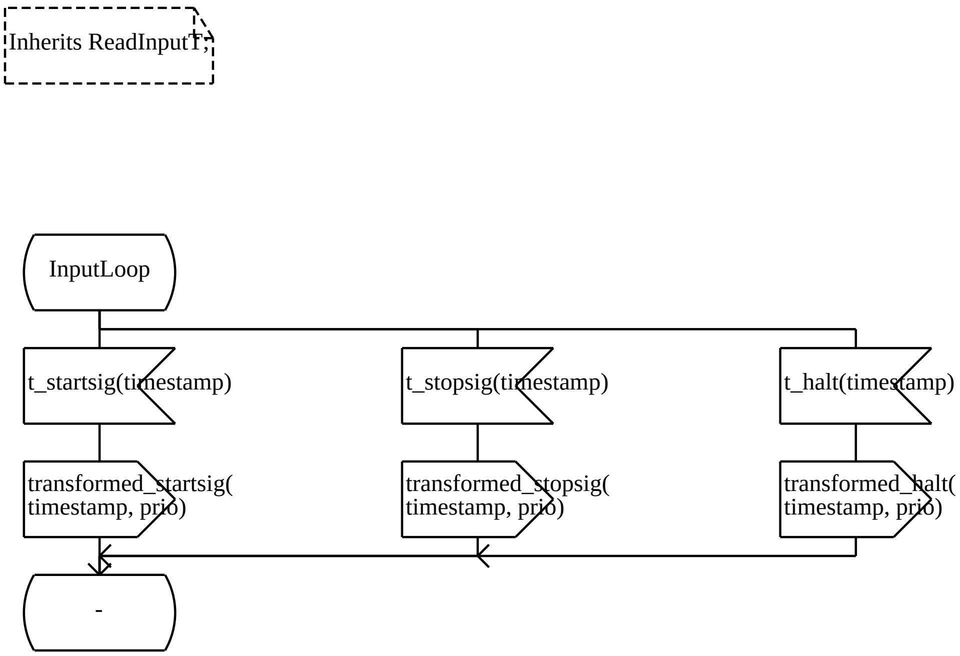

Process Type ReadHumanT
1(5)
Inherits ReadInputT;
DCL timestamp time; /* Zeit */
InputLoop
t_init(timestamp)
t_endinit
transformed_init(
timestamp, prio)
transformed_endinit(
timestamp, prio)
-
ReadInT
(t_input_stream)

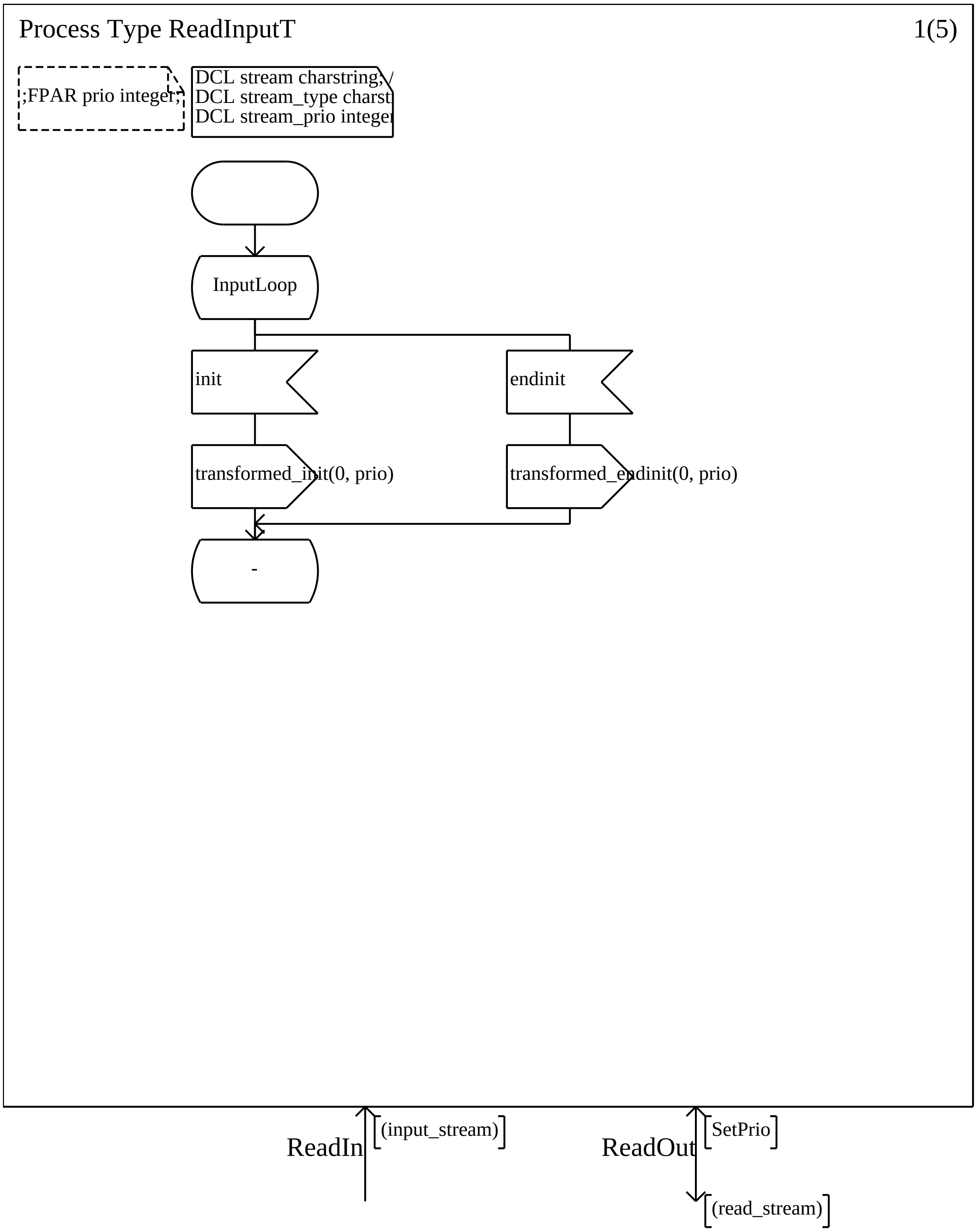


Process Type ReadHumanT
2(5)
Inherits ReadInputT;
InputLoop
t_init_in_stream(timestamp, stream, stream_type, stream_prio)
t_init_out_stream(timestamp, stream, stream_type)
t_init_periodic(timestamp, periodicity, obj_name, var_name, out_stream)
transformed_init_in_stream(timestamp, stream, stream_type, stream_prio, prio)
transformed_init_out_stream(timestamp, stream, stream_type, prio)
transformed_init_periodic(timestamp, periodicity, obj_name, var_name, out_stream, prio)
-
InputLoop
t_init_state_change(timestamp, obj_name, var_name, out_stream)
t_init_sim_interval(timestamp, start_time, stop_time)
t_init_slack(timestamp, slack)
transformed_init_state_change(timestamp, var_name, obj_name, out_stream, prio)
transformed_init_sim_interval(timestamp, start_time, stop_time, prio)
transformed_init_slack(timestamp, slack, prio)
-


Inherits ReadInputT;
InputLoop
t_init_timing_factor(timestamp, timing_factor)
t_init_logging_flag(timestamp, logging_flag)
transformed_init_timing_factor(timestamp, timing_factor, prio)
transformed_init_logging_flag(timestamp, logging_flag, prio)
-

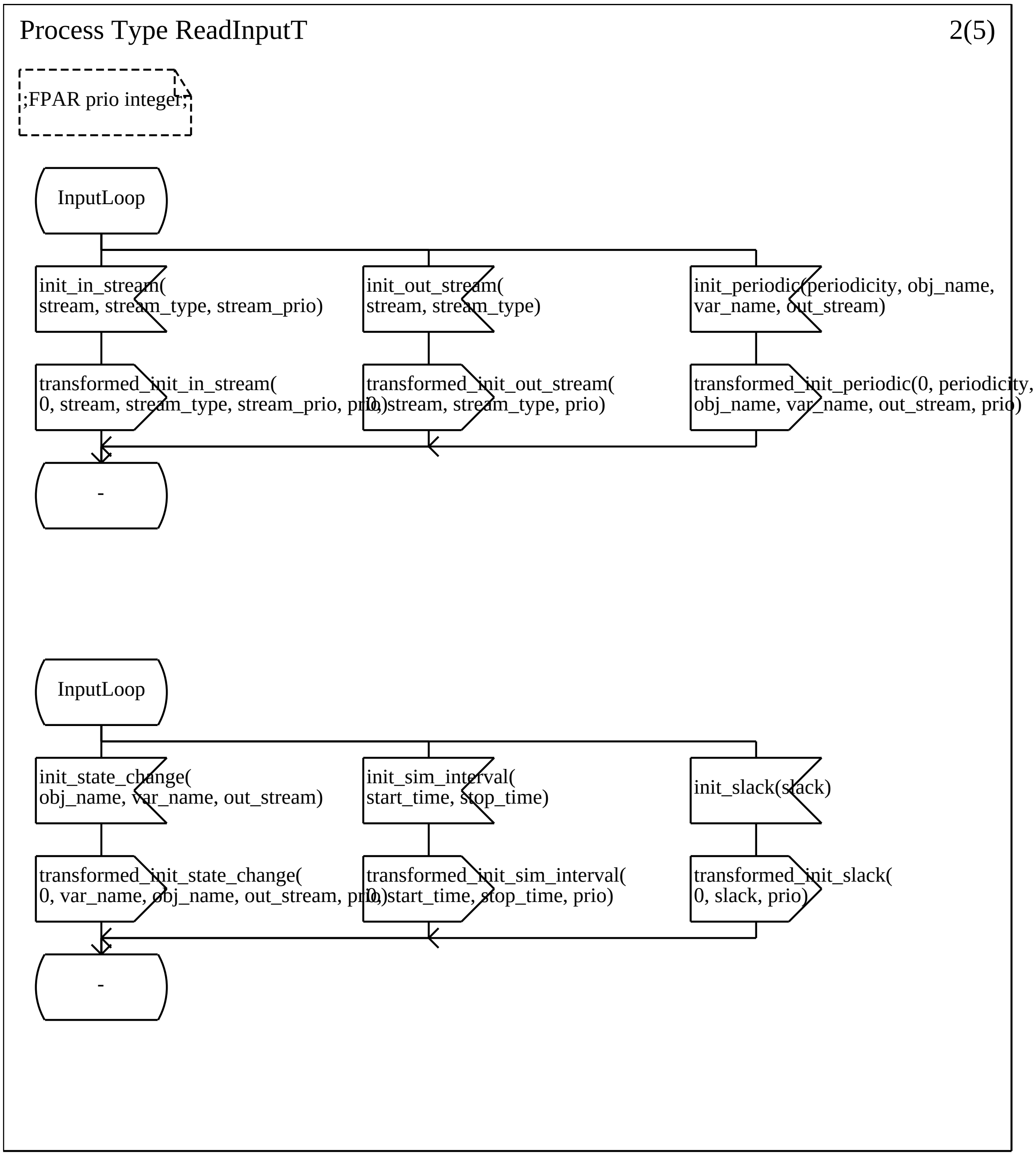

Process Type ReadHumanT
4(5)
Inherits ReadInputT;
InputLoop
t_poll(timestamp, poll_type,
obj_name, var_name, out_stream)
t_set_value(timestamp,
obj_name, var_name, value)
transformed_poll(timestamp, poll_type,
obj_name, var_name, out_stream, prio)
transformed_set_value(timestamp,
obj_name, var_name, value, prio)
-

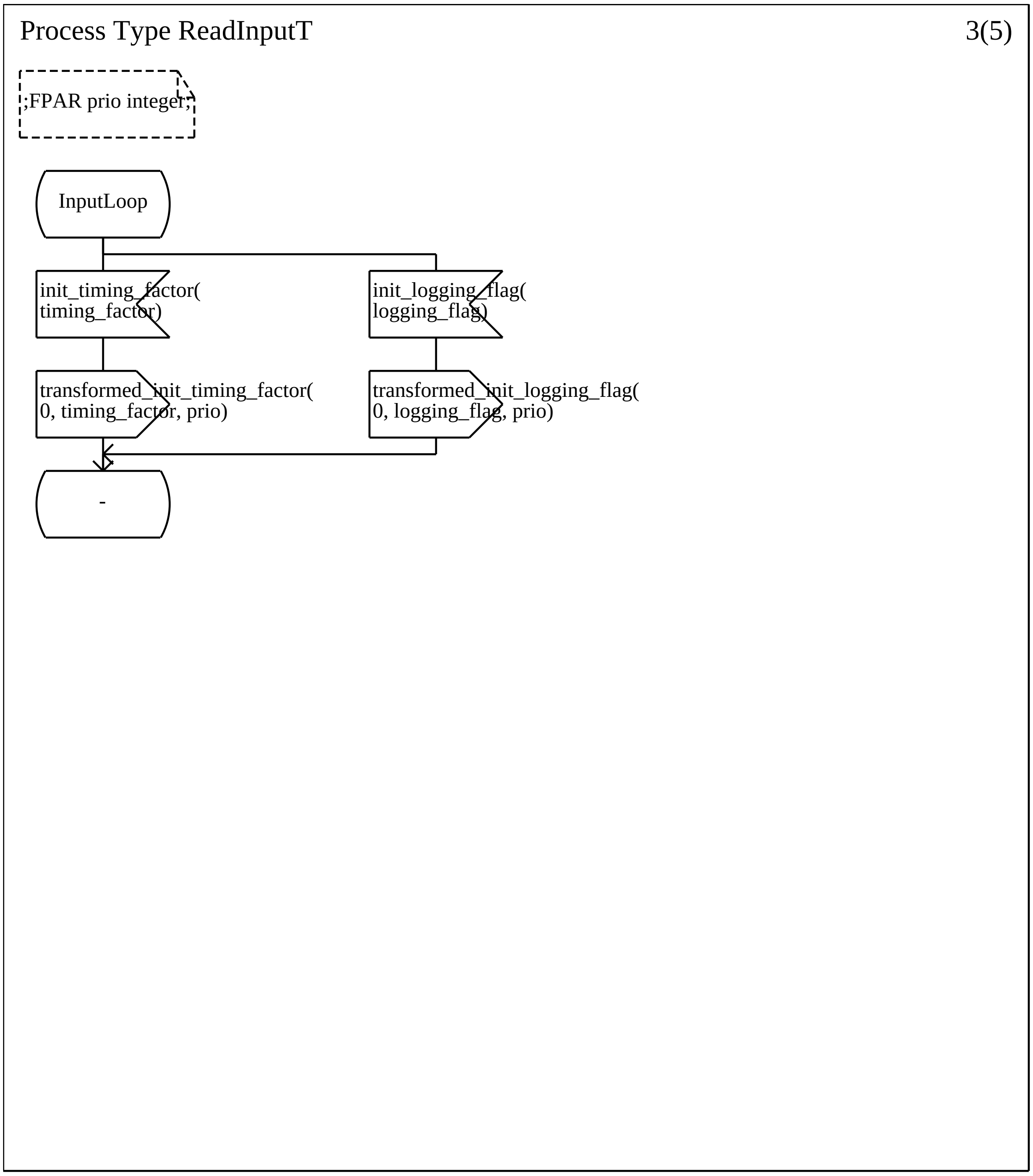

Inherits ReadInputT;
InputLoop
t_startsig(timestamp)
t_stopsig(timestamp)
t_halt(timestamp)
transformed_startsig(
timestamp, prio)
transformed_stopsig(
timestamp, prio)
transformed_halt(
timestamp, prio)
-



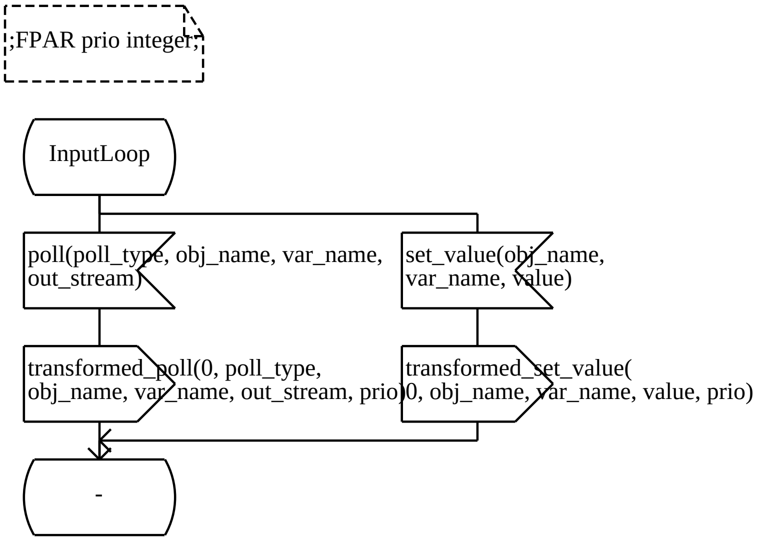
Process Type ReadInputT
1(5)
;FPAR prio integer;
DCL stream charstring,
DCL stream_type charst
DCL stream_prio intege
InputLoop
init
endinit
transformed_init(0, prio)
transformed_endinit(0, prio)
-
ReadIn
(input_stream)
ReadOut
SetPrio
(read_stream)

```
DCL stream charstring; /* Stream-Name */
DCL stream_type charstring; /* Stream-Typ */
DCL stream_prio integer; /* Stream-Priority */

DCL obj_name charstring; /* Objectname */
DCL var_name charstring; /* Variablenname */
DCL value charstring;        /* Wert */
DCL poll_type charstring; /* Abfragetyp */
DCL out_stream charstring; /* Stream fuer Ausgabe */

DCL periodicity duration;   /* Frequenz der regelmaessigen Ausgabe */
DCL start_time time; /* Anfangszeit der Simulation */
DCL stop_time time;     /* End der Simulation */
DCL slack time;         /* Maxiamle Zeitabweichnung */
DCL timing_factor real; /* Quotient zwischen reeler und simulierter Zeit */
DCL logging_flag boolean; /* Soll ein log geschrieben werden */
```

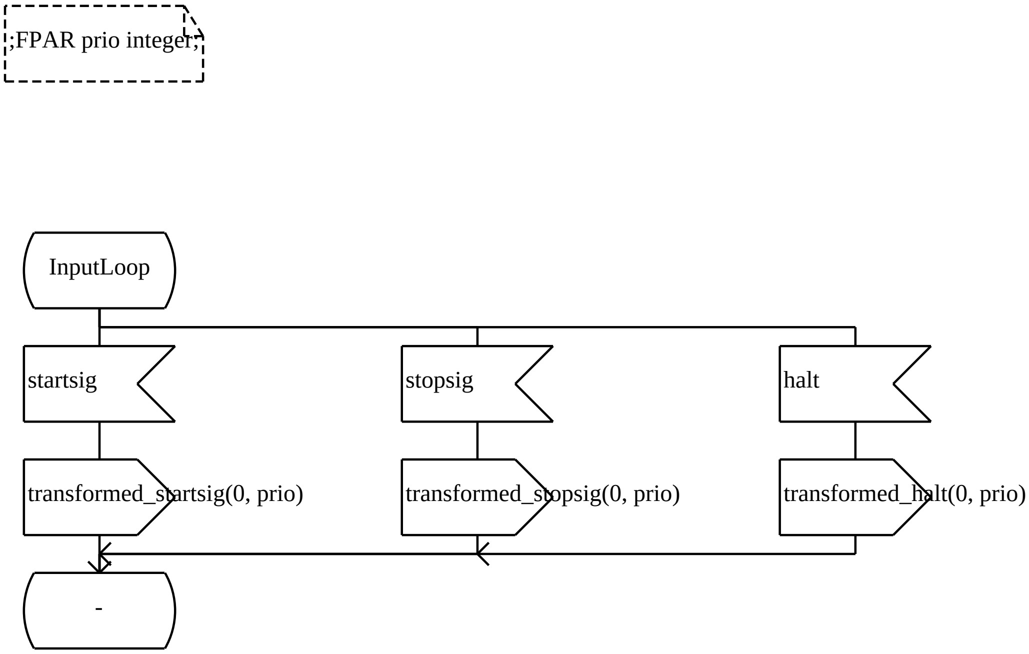

Process Type ReadInputT
2(5)
;FPAR prio integer;
InputLoop
init_in_stream(
stream, stream_type, stream_prio)
init_out_stream(
stream, stream_type)
init_periodic(periodicity, obj_name,
var_name, out_stream)
transformed_init_in_stream(
0, stream, stream_type, stream_prio, prio)
transformed_init_out_stream(
0, stream, stream_type, prio)
transformed_init_periodic(0, periodicity,
obj_name, var_name, out_stream, prio)
-
InputLoop
init_state_change(
obj_name, var_name, out_stream)
init_sim_interval(
start_time, stop_time)
init_slack(slack)
transformed_init_state_change(
0, var_name, obj_name, out_stream, prio)
transformed_init_sim_interval(
0, start_time, stop_time, prio)
transformed_init_slack(
0, slack, prio)
-


Process Type ReadInputT
3(5)
;FPAR prio integer;
InputLoop
init_timing_factor(
timing_factor)
init_logging_flag(
logging_flag)
transformed_init_timing_factor(
0, timing_factor, prio)
transformed_init_logging_flag(
0, logging_flag, prio)
-

Process Type ReadInputT



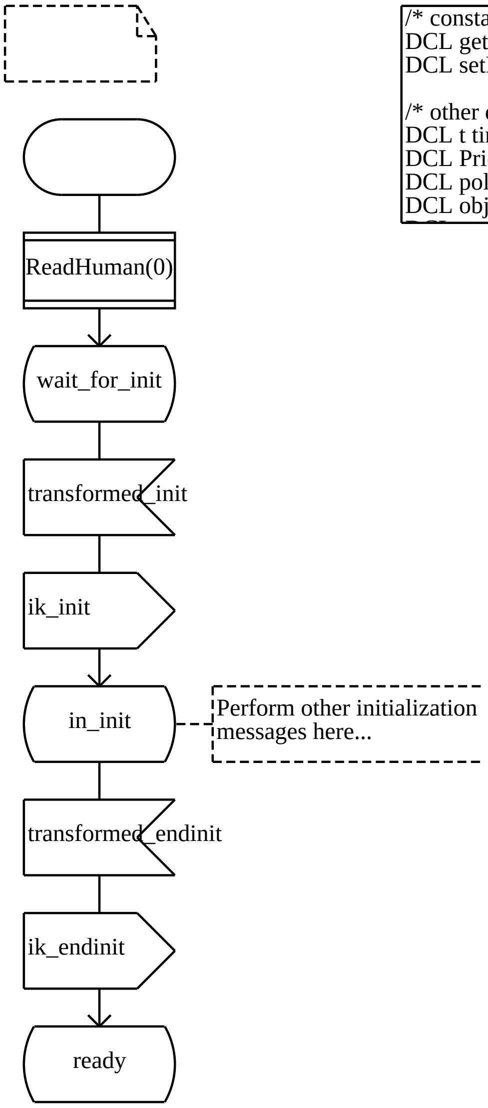
Process Type ReadInputT



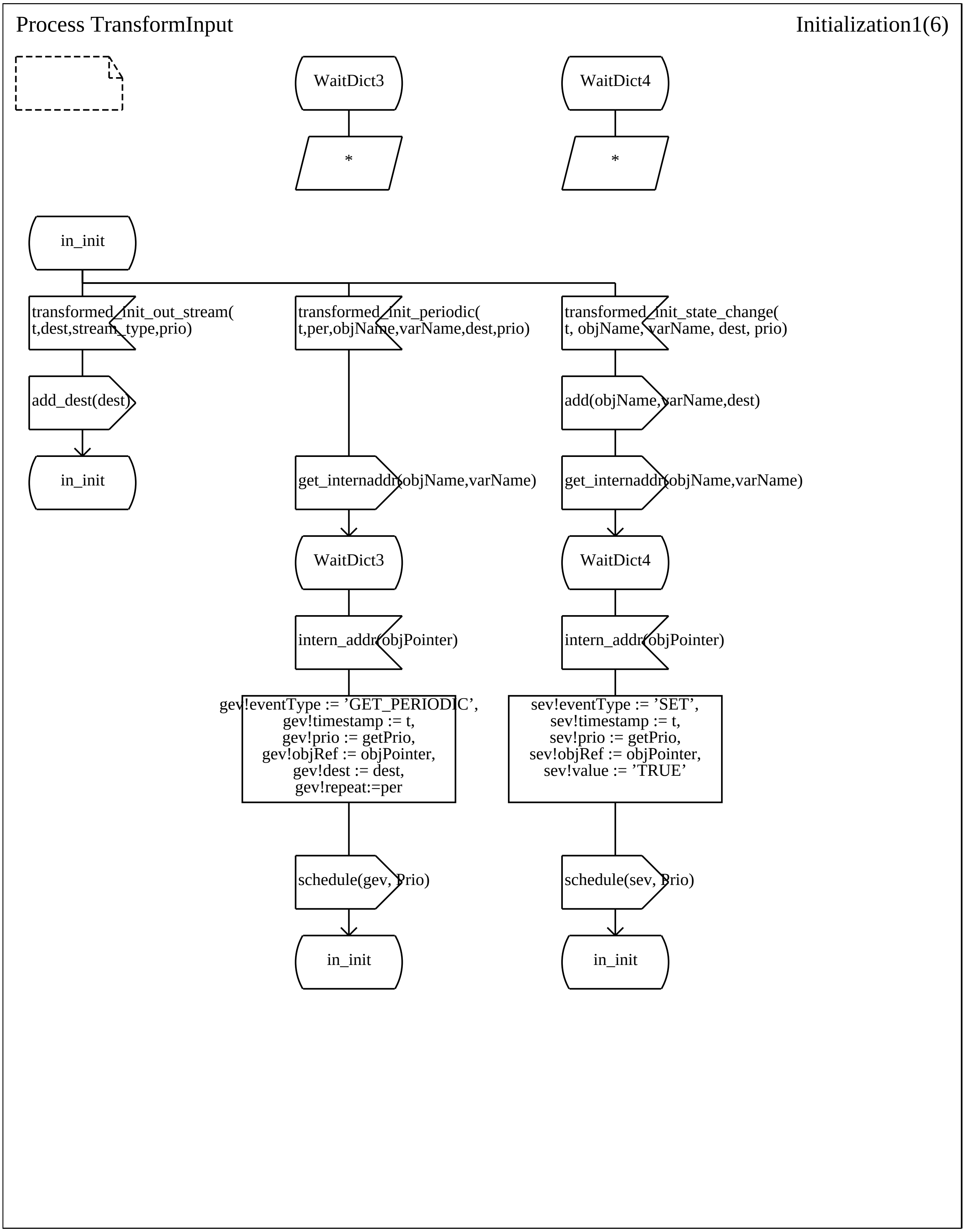
Process TransformInput

Initialization(6)

```
/* constants */
DCL getPrio integer:=1;
DCL setPrio integer:=1;

/* other declarations */
DCL t time; /* when to execute... */
DCL Prio integer; /* Priority */
DCL pollType charstring; /* usually 'VALUE' */
DCL objName charstring;
```

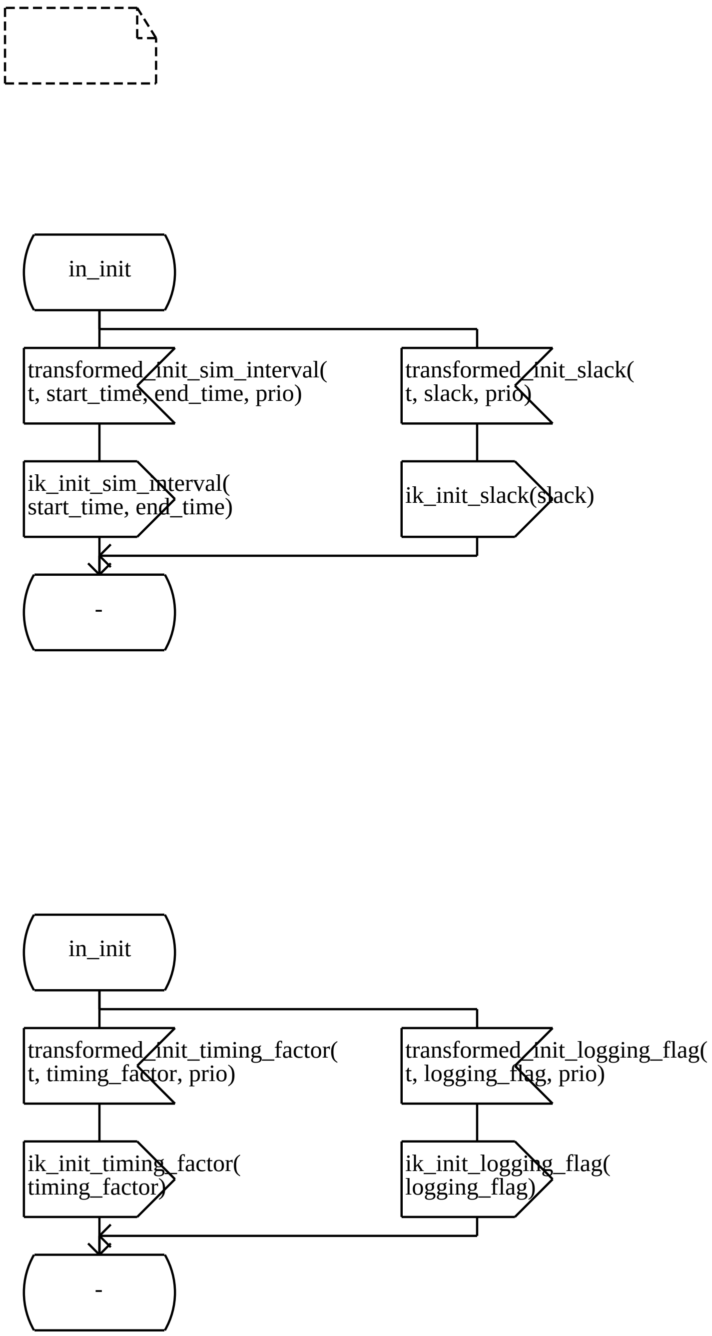


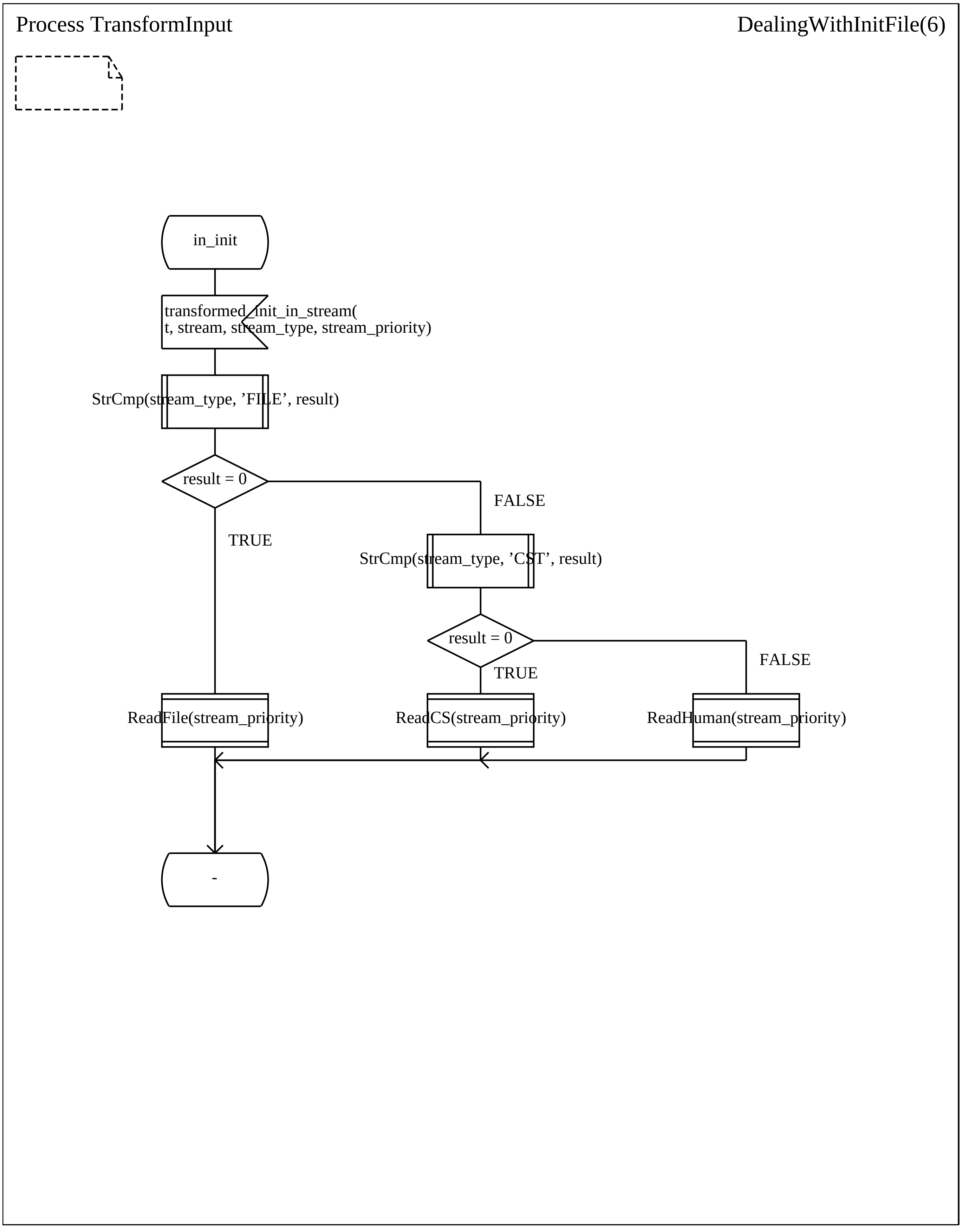
```
/* constants */
DCL getPrio integer:=1;
DCL setPrio integer:=1;

/* other declarations */
DCL t time; /* when to execute... */
DCL Prio integer; /* Priority */
DCL pollType charstring; /* usually 'VALUE' */
DCL objName charstring;
DCL varName charstring;
DCL newValue charstring;
DCL dest charstring; /* output stream */
DCL per duration; /* repeat rate */

DCL objPointer integer; /* Returned from Dictionary */
/* DCL gev getEvent; *//* Event */
DCL gev event; /* Event */
DCL sev event; /* Event */

DCL stream charstring; /* Name des Stream */
DCL stream_type charstring; /* Art des Eingabestream */
DCL stream_priority integer; /* Prioritaet des Streams */

DCL result integer; /* Ergebnis von strcmp */

/* Fuer Kernel-Init */
DCL start_time    time;
DCL end_time      time;
DCL slack         time;
DCL timing_factor real;
DCL logging_flag  boolean;
```

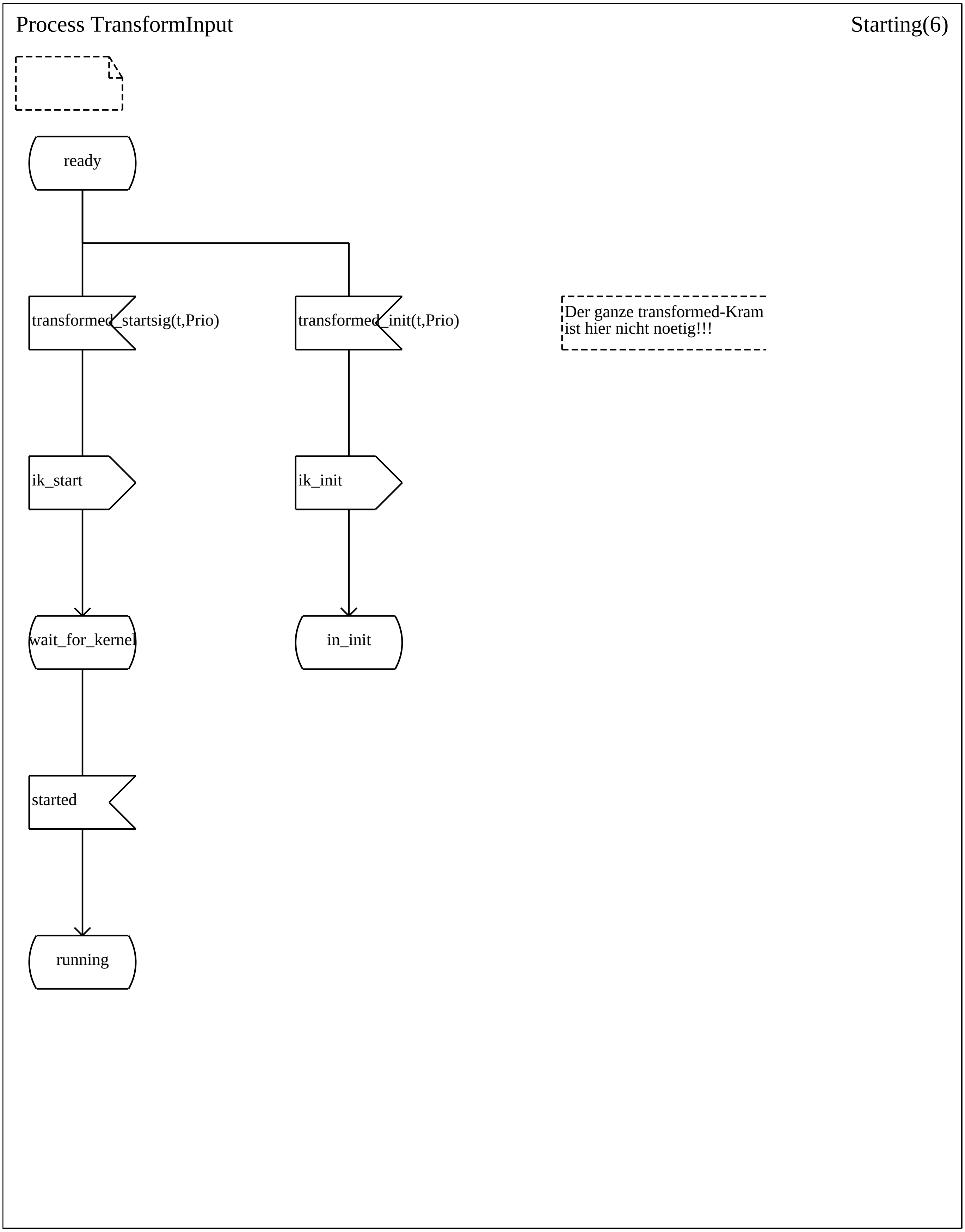

Process TransformInput
Initialization1(6)
WaitDict3
*
WaitDict4
*
in_init
transformed_init_out_stream(
t,dest,stream_type,prio)
add_dest(dest)
in_init
transformed_init_periodic(
t,per,objName,varName,dest,prio)
get_internaddr(objName,varName)
WaitDict3
intern_addr(objPointer)
gev!eventType := 'GET_PERIODIC',
gev!timestamp := t,
gev!prio := getPrio,
gev!objRef := objPointer,
gev!dest := dest,
gev!repeat:=per
schedule(gev, Prio)
in_init
transformed_init_state_change(
t, objName, varName, dest, prio)
add(objName,varName,dest)
get_internaddr(objName,varName)
WaitDict4
intern_addr(objPointer)
sev!eventType := 'SET',
sev!timestamp := t,
sev!prio := getPrio,
sev!objRef := objPointer,
sev!value := 'TRUE'
schedule(sev, Prio)
in_init

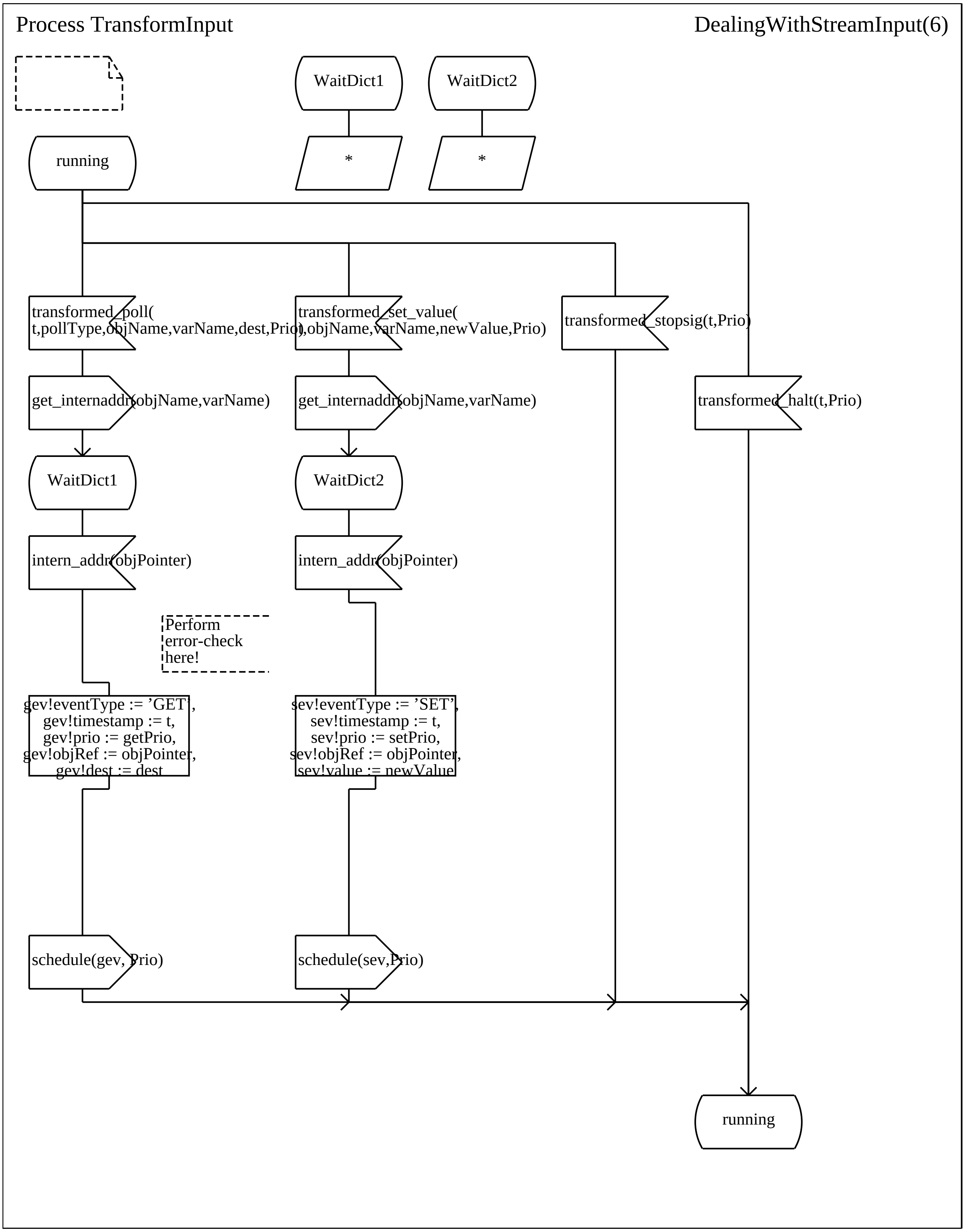
Process TransformInput

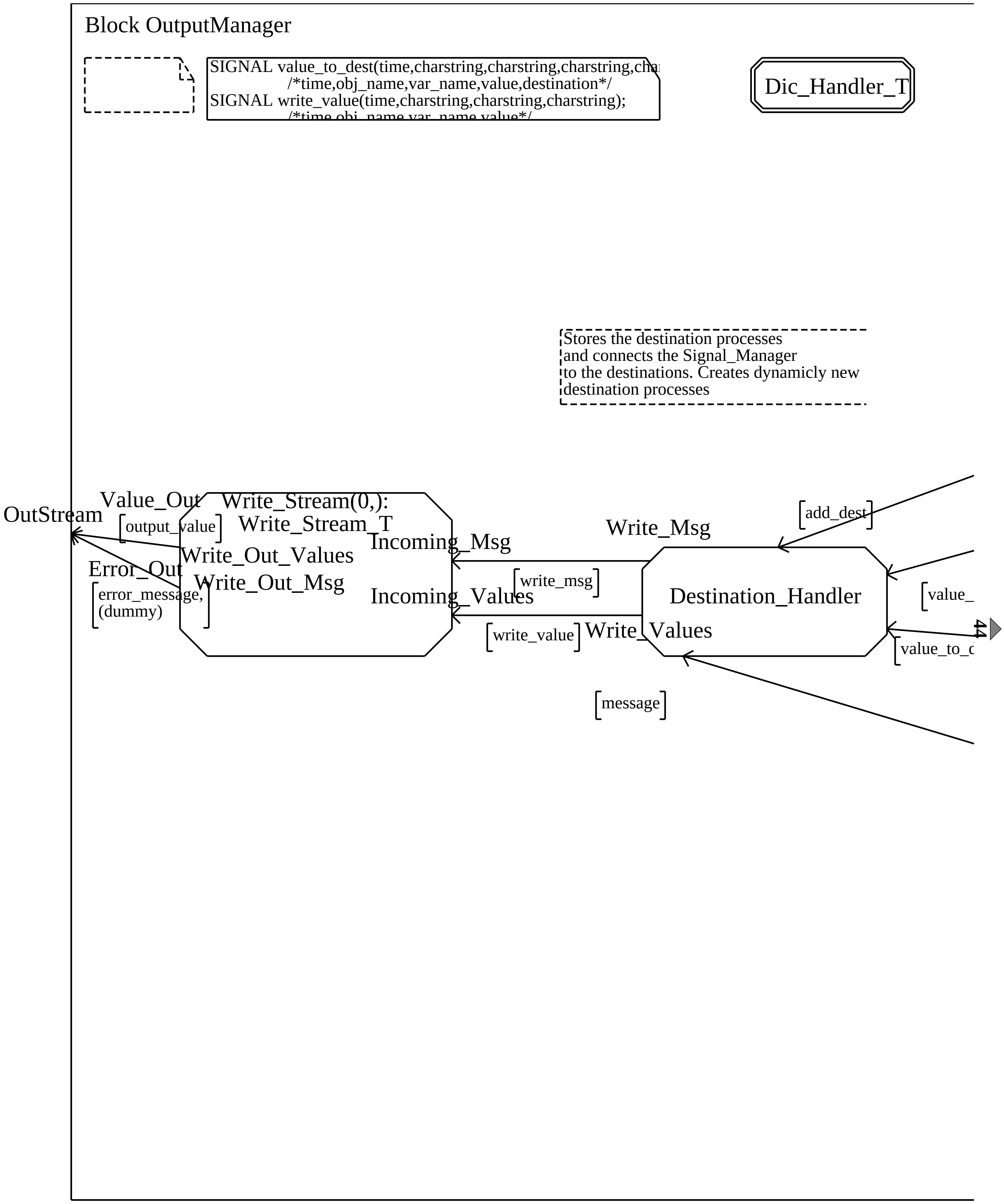
Initialization_Kernel(6)

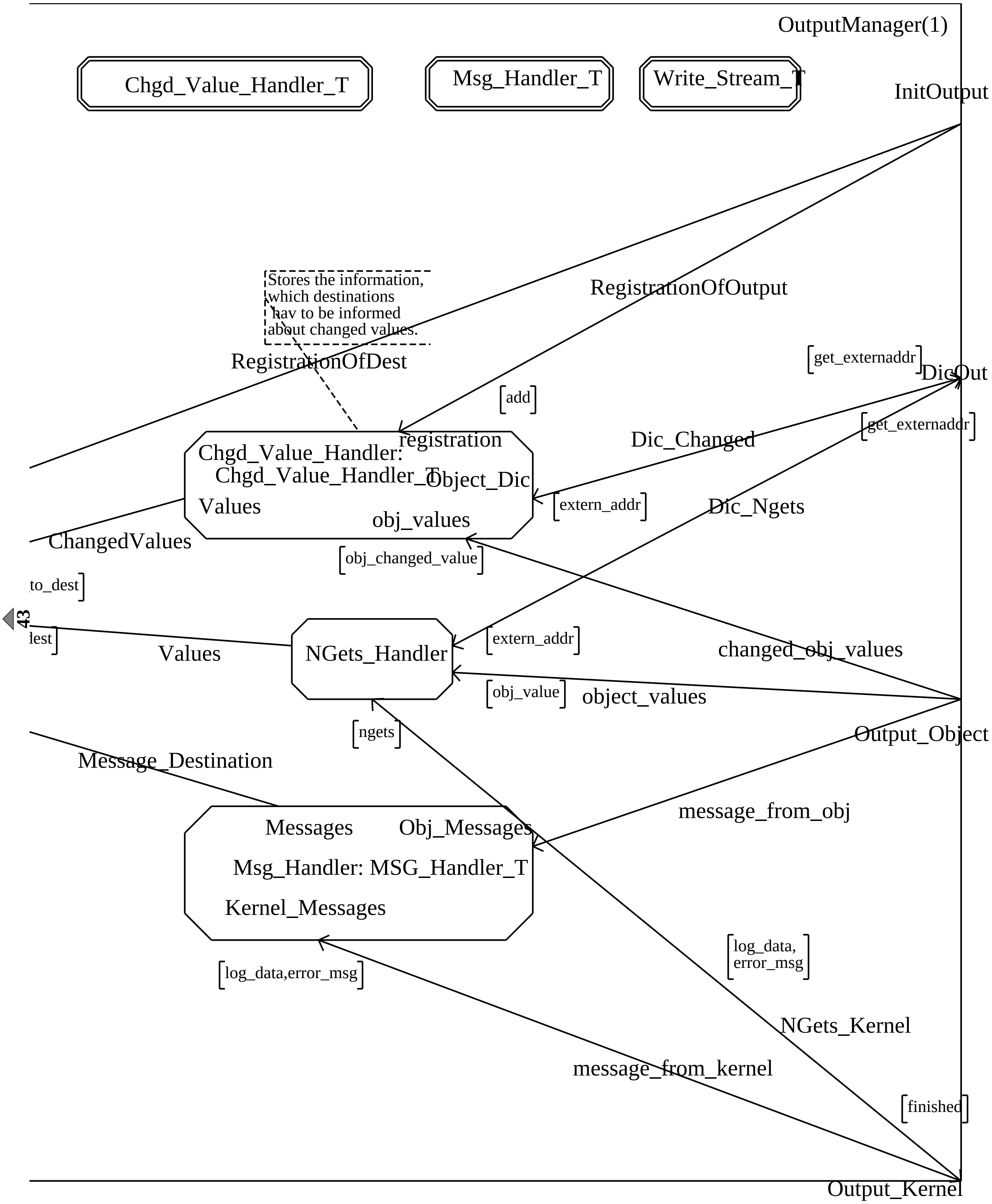



Process TransformInput
DealingWithInitFile(6)
in_init
transformed_init_in_stream(
t, stream, stream_type, stream_priority)
StrCmp(stream_type, 'FILE', result)
result = 0
FALSE
TRUE
StrCmp(stream_type, 'CST', result)
result = 0
FALSE
TRUE
ReadFile(stream_priority)
ReadCS(stream_priority)
ReadHuman(stream_priority)
-


Process TransformInput
Starting(6)
ready
transformed_startsig(t,Prio)
transformed_init(t,Prio)
Der ganze transformed-Kram
ist hier nicht noetig!!!
ik_start
ik_init
wait_for_kernel
in_init
started
running

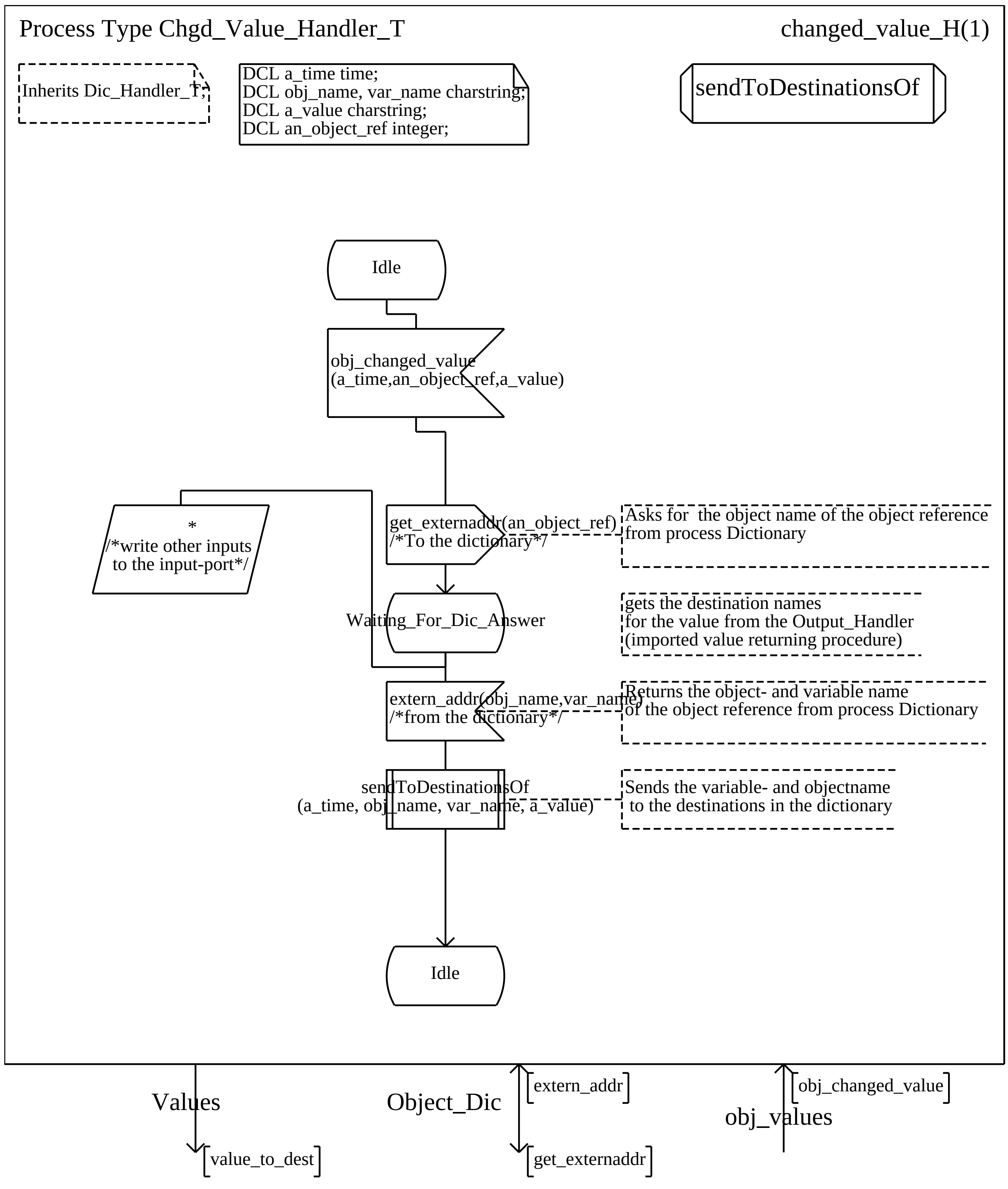

Process TransformInput
DealingWithStreamInput(6)
WaitDict1
WaitDict2
running
*
*
transformed_poll(
t,pollType,objName,varName,dest,Prio)
transformed_set_value(
t,objName,varName,newValue,Prio)
transformed_stopsig(t,Prio)
get_internaddr(objName,varName)
get_internaddr(objName,varName)
transformed_halt(t,Prio)
WaitDict1
WaitDict2
intern_addr(objPointer)
intern_addr(objPointer)
Perform
error-check
here!
gev!eventType := 'GET',
gev!timestamp := t,
gev!prio := getPrio,
gev!objRef := objPointer,
gev!dest := dest
sev!eventType := 'SET',
sev!timestamp := t,
sev!prio := setPrio,
sev!objRef := objPointer,
sev!value := newValue
schedule(gev, Prio)
schedule(sev,Prio)
running


Block OutputManager
SIGNAL value_to_dest(time,charstring,charstring,charstring,cha
/*time,obj_name,var_name,value,destination*/
SIGNAL write_value(time,charstring,charstring,charstring);
/*time,obj_name,var_name,value*/
Dic_Handler_T
Stores the destination processes
and connects the Signal_Manager
to the destinations. Creates dynamicly new
destination processes
OutStream
Value_Out
output_value
Error_Out
error_message,
(dummy)
Write_Stream(0,):
Write_Stream_T
Write_Out_Values
Write_Out_Msg
Incoming_Msg
Incoming_Values
write_msg
write_value
Write_Msg
Write_Values
Destination_Handler
add_dest
value_
value_to_c
message
44

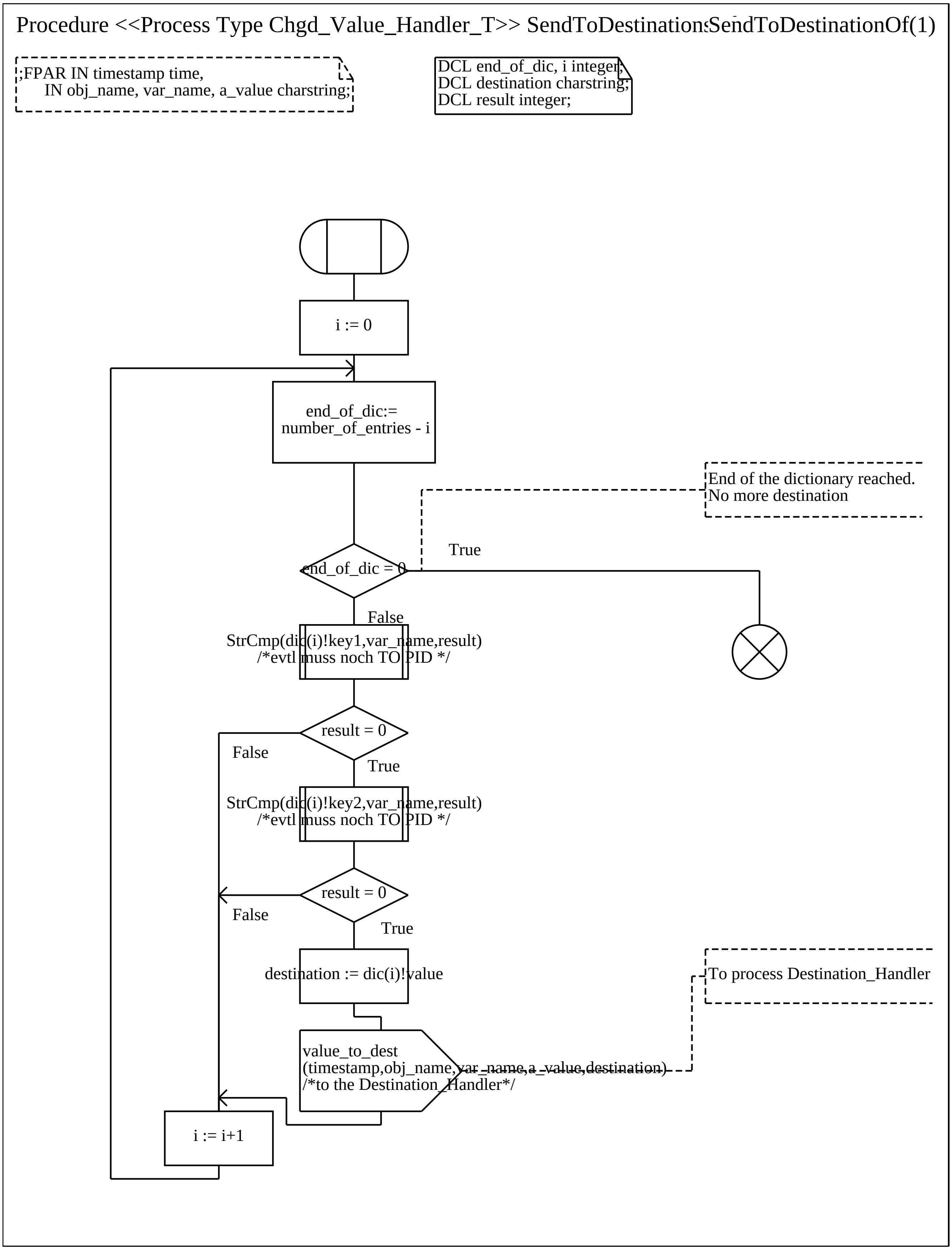


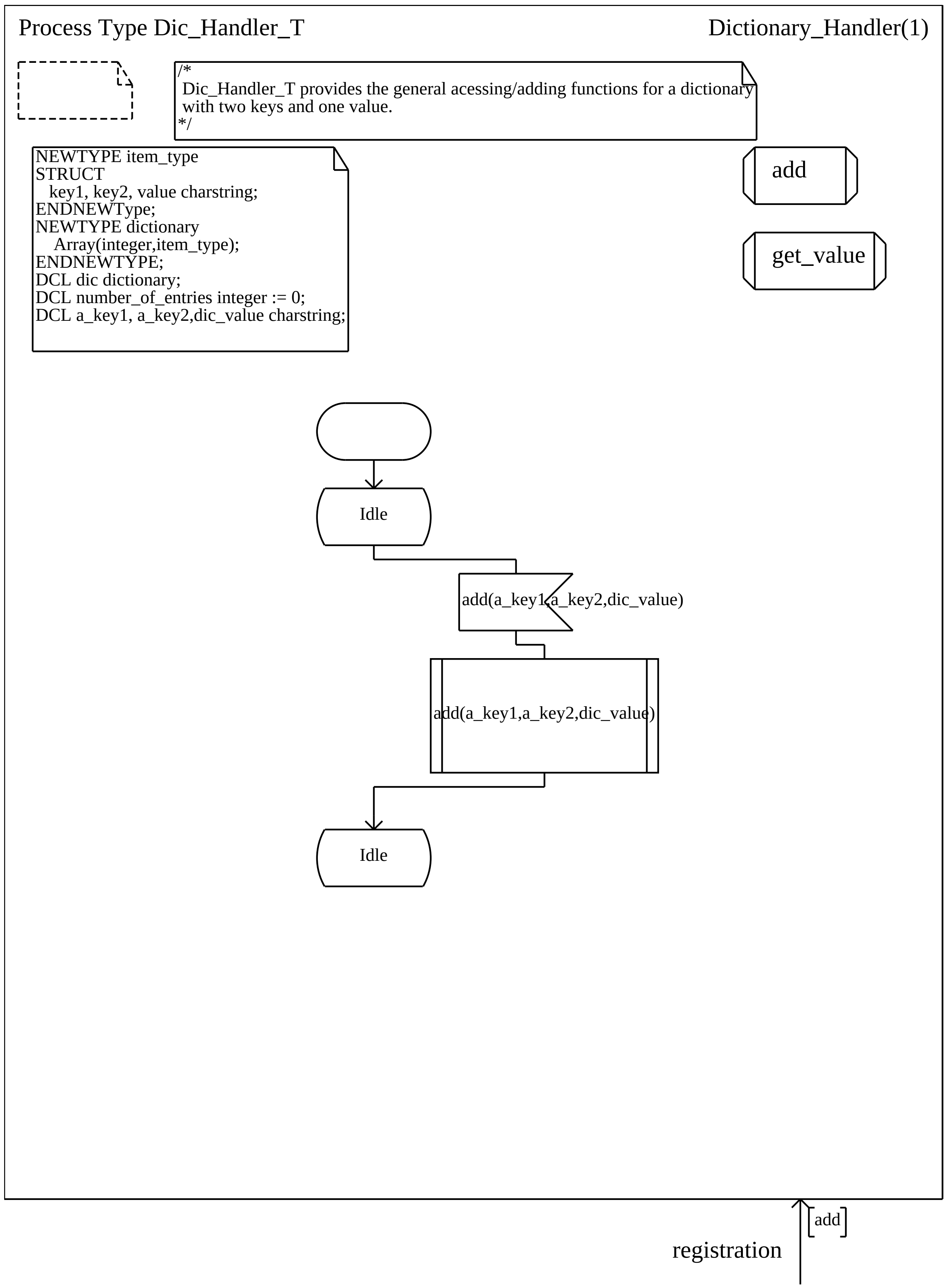
OutputManager(1)
Chgd_Value_Handler_T
Msg_Handler_T
Write_Stream_T
InitOutput
Stores the information,
which destinations
hav to be informed
about changed values.
RegistrationOfOutput
RegistrationOfDest
get_externaddr
DicOut
add
get_externaddr
registration
Dic_Changed
Chgd_Value_Handler:
Chgd_Value_Handler_T
Object_Dic
Values
extern_addr
Dic_Ngets
obj_values
ChangedValues
obj_changed_value
to_dest
43
dest
extern_addr
changed_obj_values
Values
NGets_Handler
obj_value
object_values
ngets
Output_Object
Message_Destination
message_from_obj
Messages
Obj_Messages
Msg_Handler: MSG_Handler_T
Kernel_Messages
log_data,
error_msg
log_data,error_msg
NGets_Kernel
message_from_kernel
finished
Output_Kernel

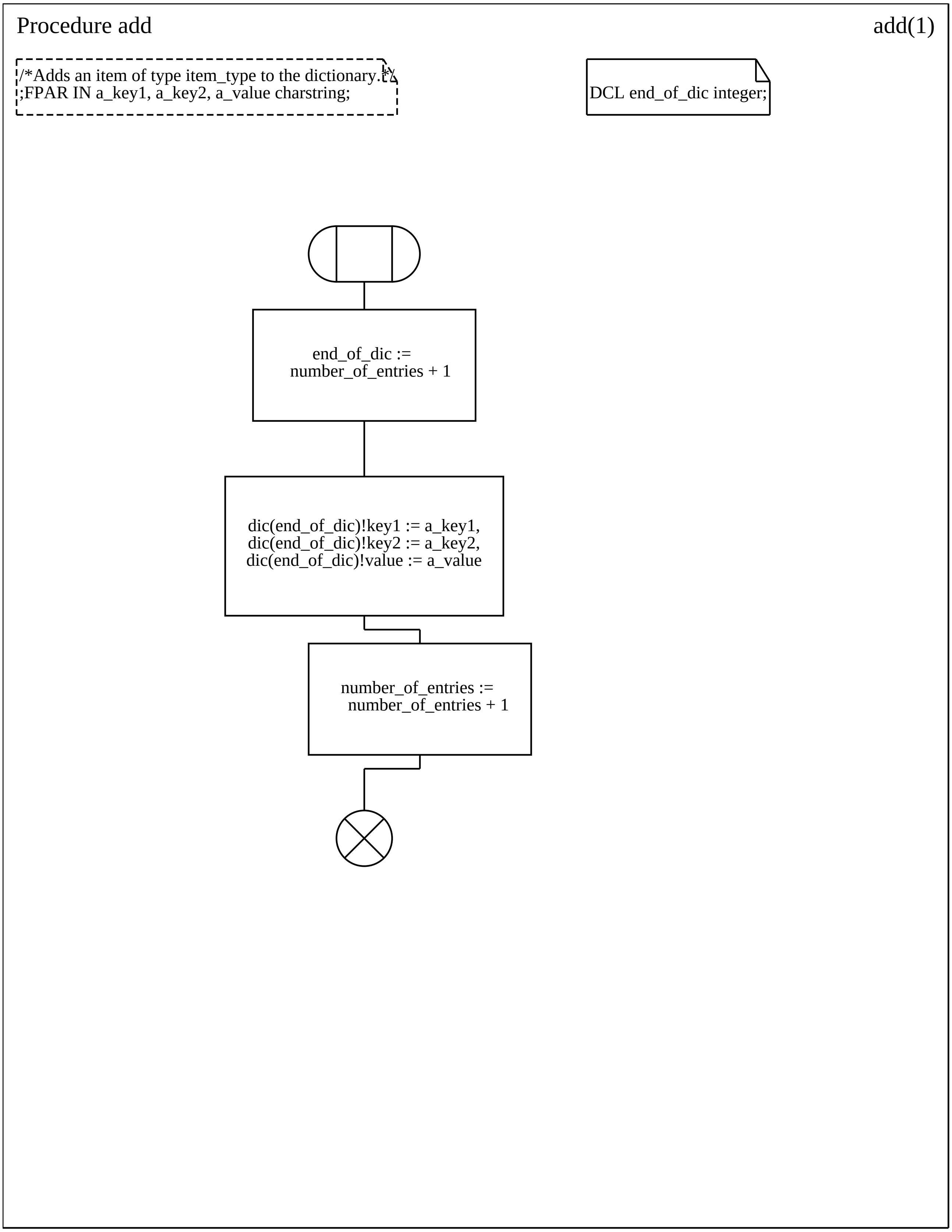
```
SIGNAL value_to_dest(time,charstring,charstring,charstring,charstring);
        /*time,obj_name,var_name,value,destination*/
SIGNAL write_value(time,charstring,charstring,charstring);
        /*time,obj_name,var_name,value*/
SIGNAL write_msg(charstring);
SIGNAL message(charstring);
SIGNALLIST dummy = result, ack, log;
```

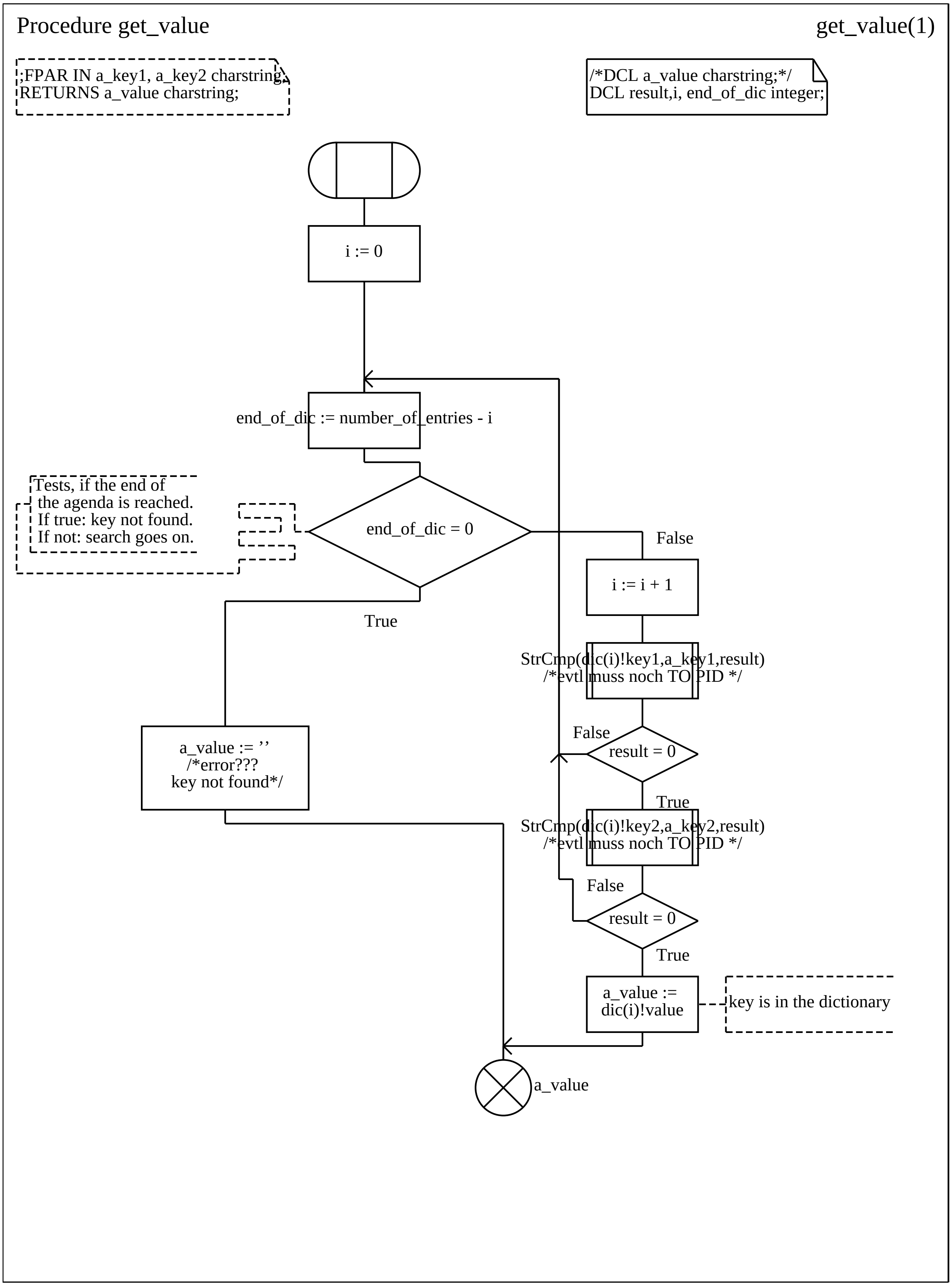


Process Type Chgd_Value_Handler_T
changed_value_H(1)
Inherits Dic_Handler_T;
DCL a_time time;
DCL obj_name, var_name charstring;
DCL a_value charstring;
DCL an_object_ref integer;
sendToDestinationsOf
Idle
obj_changed_value
(a_time,an_object_ref,a_value)
*
/*write other inputs
to the input-port*/
get_externaddr(an_object_ref)
/*To the dictionary*/
Asks for the object name of the object reference
from process Dictionary
Waiting_For_Dic_Answer
gets the destination names
for the value from the Output_Handler
(imported value returning procedure)
extern_addr(obj_name,var_name)
/*from the dictionary*/
Returns the object- and variable name
of the object reference from process Dictionary
sendToDestinationsOf
(a_time, obj_name, var_name, a_value)
Sends the variable- and objectname
to the destinations in the dictionary
Idle
Values
value_to_dest
Object_Dic
extern_addr
get_externaddr
obj_values
obj_changed_value

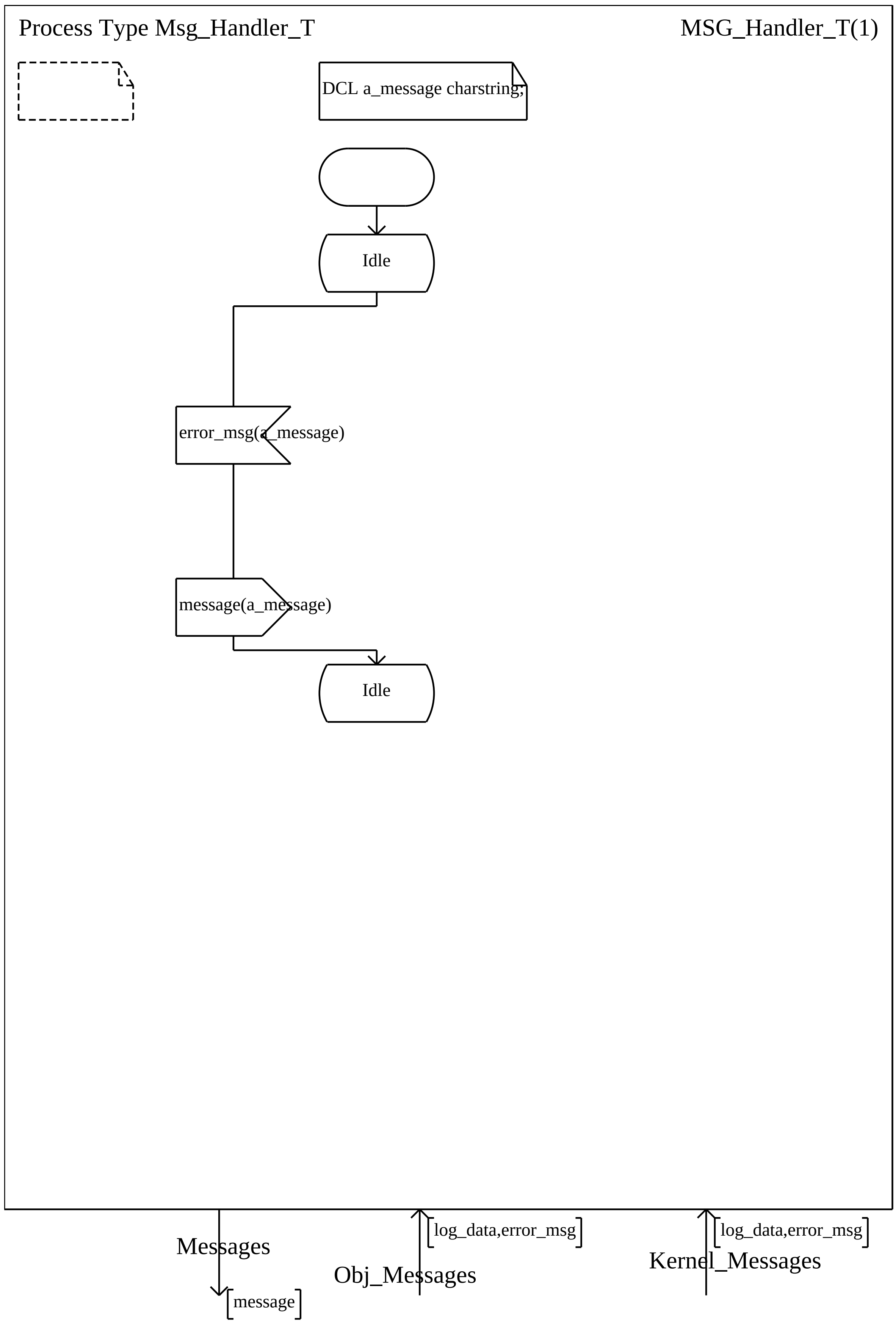

Procedure <<Process Type Chgd_Value_Handler_T>> SendToDestinationsSendToDestinationOf(1)
;FPAR IN timestamp time,
IN obj_name, var_name, a_value charstring;
DCL end_of_dic, i integer;
DCL destination charstring;
DCL result integer;
i := 0
end_of_dic:=
number_of_entries - i
End of the dictionary reached.
No more destination
True
end_of_dic = 0
False
StrCmp(dic(i)!key1,var_name,result)
/*evtl muss noch TO PID */
result = 0
False
True
StrCmp(dic(i)!key2,var_name,result)
/*evtl muss noch TO PID */
result = 0
False
True
destination := dic(i)!value
To process Destination_Handler
value_to_dest
(timestamp,obj_name,var_name,a_value,destination)
/*to the Destination_Handler*/
i := i+1

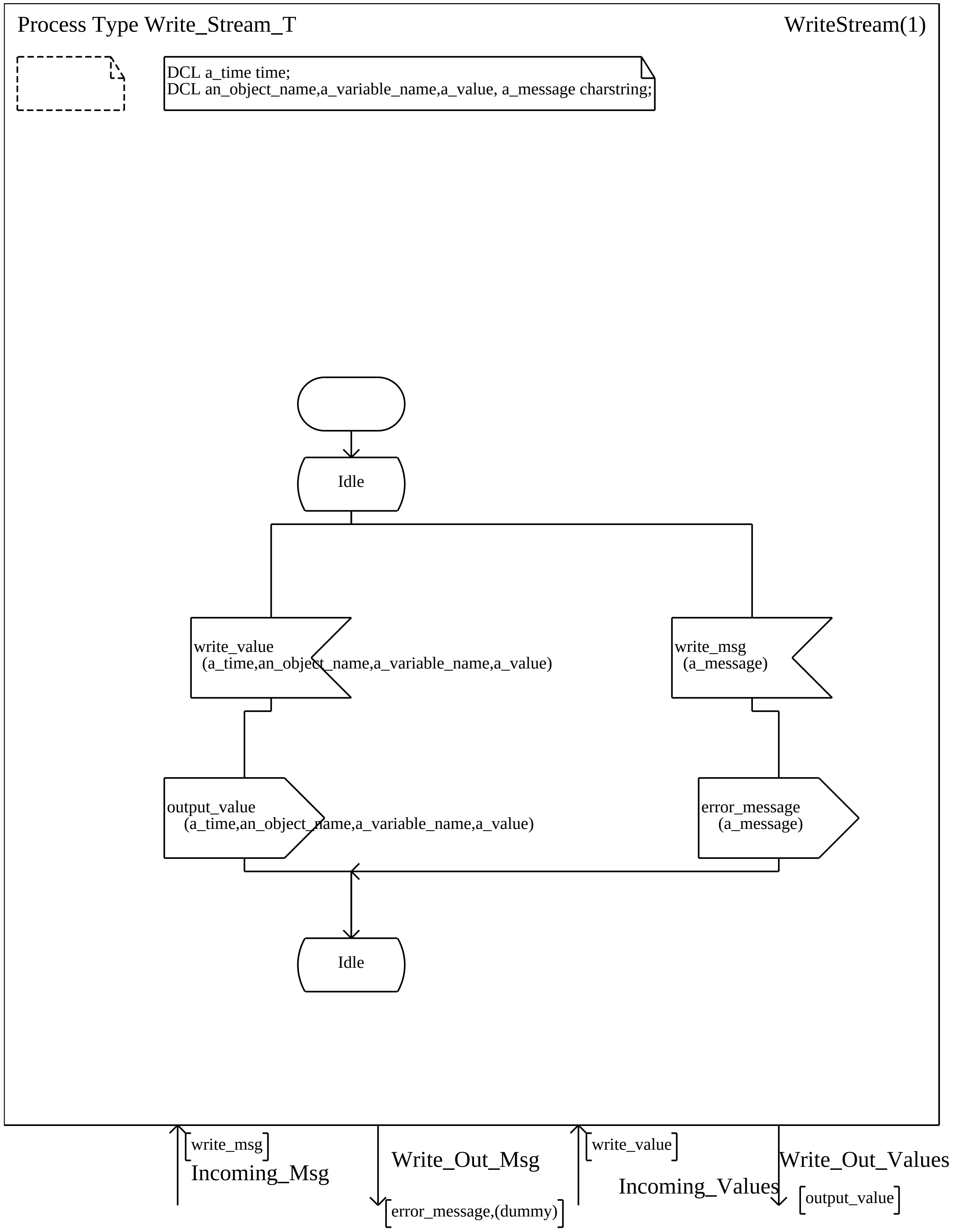

Process Type Dic_Handler_T
Dictionary_Handler(1)
/*
Dic_Handler_T provides the general acessing/adding functions for a dictionary
with two keys and one value.
*/
NEWTYPE item_type
STRUCT
key1, key2, value charstring;
ENDNEWType;
NEWTYPE dictionary
Array(integer,item_type);
ENDNEWTYPE;
DCL dic dictionary;
DCL number_of_entries integer := 0;
DCL a_key1, a_key2,dic_value charstring;
add
get_value
Idle
add(a_key1,a_key2,dic_value)
add(a_key1,a_key2,dic_value)
Idle
registration
[add]

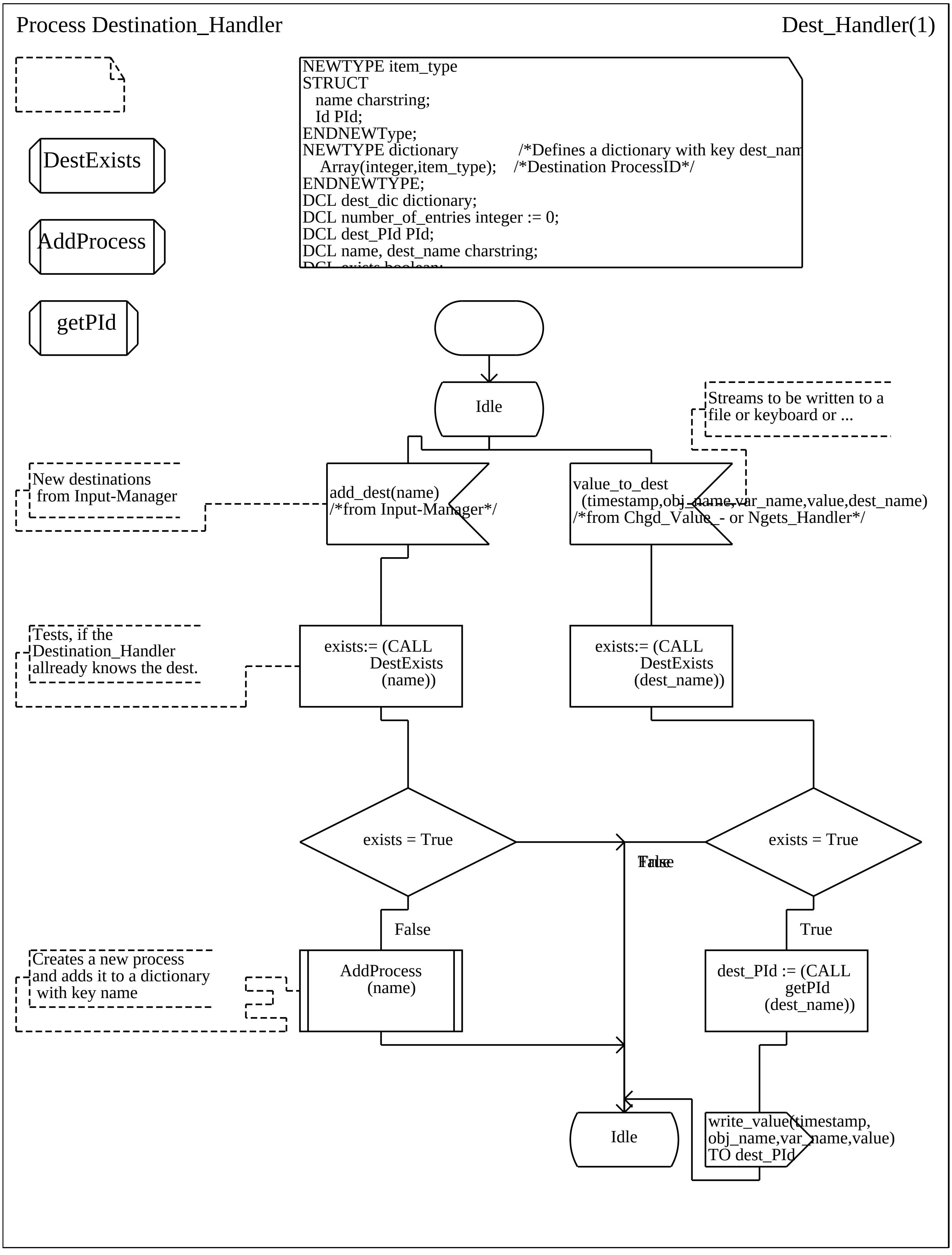

Procedure add
add(1)
/*Adds an item of type item_type to the dictionary.*/
;FPAR IN a_key1, a_key2, a_value charstring;
DCL end_of_dic integer;
end_of_dic :=
number_of_entries + 1
dic(end_of_dic)!key1 := a_key1,
dic(end_of_dic)!key2 := a_key2,
dic(end_of_dic)!value := a_value
number_of_entries :=
number_of_entries + 1



Procedure get_value
get_value(1)
;FPAR IN a_key1, a_key2 charstring;
RETURNS a_value charstring;
/*DCL a_value charstring;*/
DCL result,i, end_of_dic integer;
i := 0
end_of_dic := number_of_entries - i
Tests, if the end of
the agenda is reached.
If true: key not found.
If not: search goes on.
end_of_dic = 0
False
True
i := i + 1
StrCmp(dic(i)!key1,a_key1,result)
/*evtl muss noch TO PID */
False
result = 0
True
StrCmp(dic(i)!key2,a_key2,result)
/*evtl muss noch TO PID */
False
result = 0
True
a_value := ’’
/*error???
key not found*/
a_value :=
dic(i)!value
key is in the dictionary
a_value

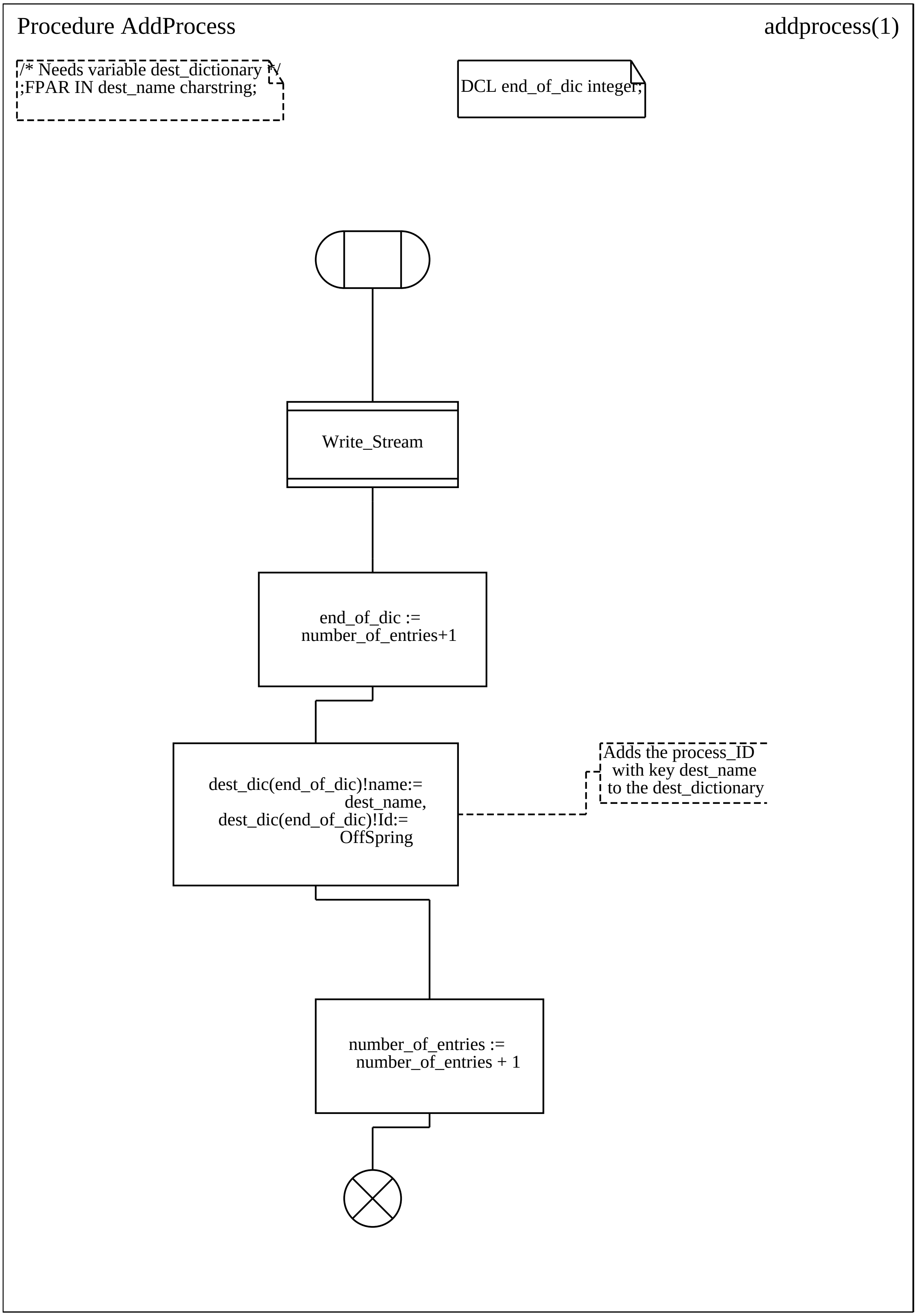

Process Type Msg_Handler_T
MSG_Handler_T(1)
DCL a_message charstring;
Idle
error_msg(a_message)
message(a_message)
Idle
Messages
message
Obj_Messages
log_data,error_msg
Kernel_Messages
log_data,error_msg

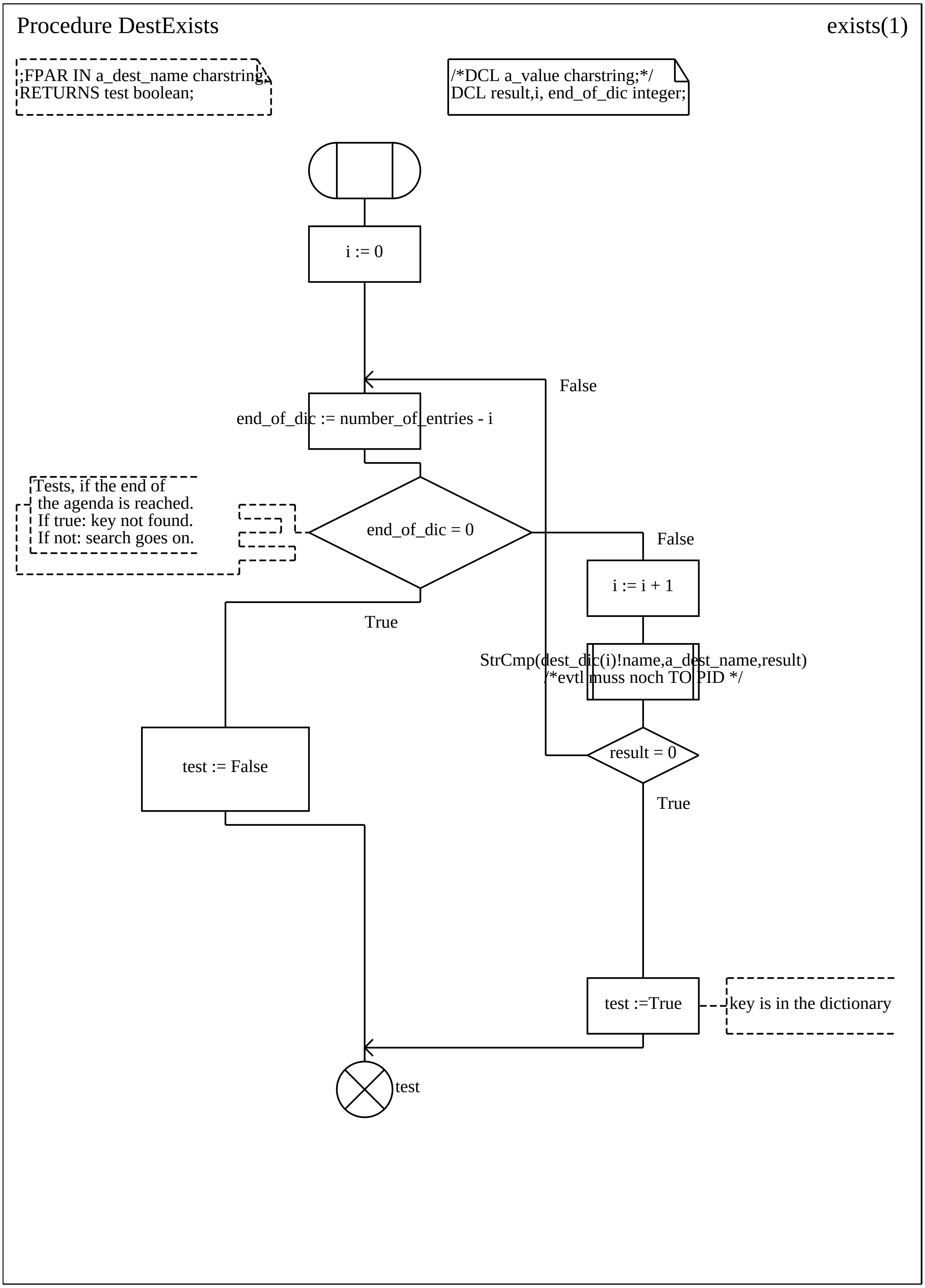

Process Type Write_Stream_T
WriteStream(1)
DCL a_time time;
DCL an_object_name,a_variable_name,a_value, a_message charstring;
Idle
write_value
(a_time,an_object_name,a_variable_name,a_value)
write_msg
(a_message)
output_value
(a_time,an_object_name,a_variable_name,a_value)
error_message
(a_message)
Idle
write_msg
Incoming_Msg
Write_Out_Msg
error_message,(dummy)
write_value
Incoming_Values
Write_Out_Values
output_value



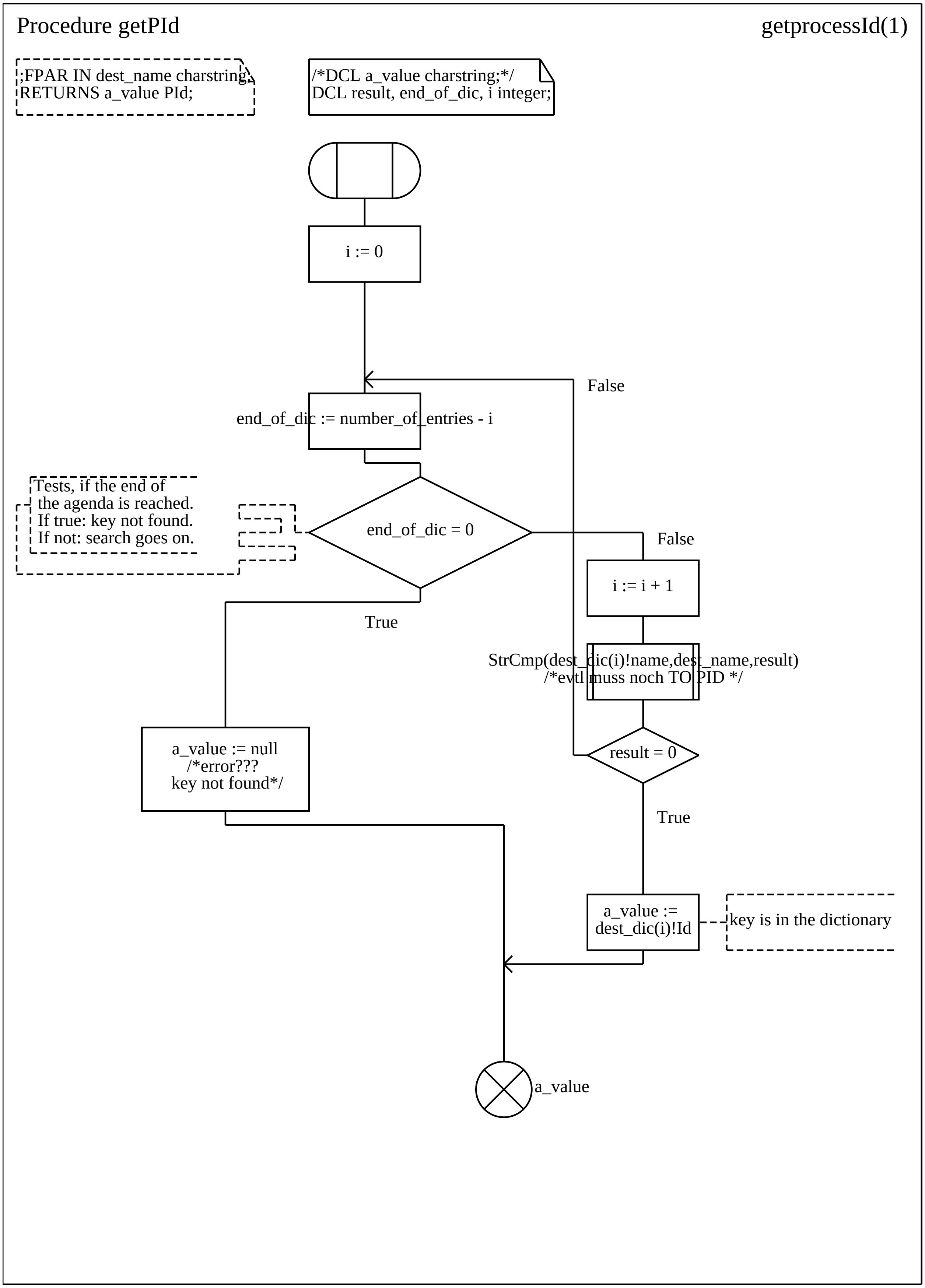
Process Destination_Handler
Dest_Handler(1)
DestExists
AddProcess
getPId
NEWTYPE item_type
STRUCT
name charstring;
Id PId;
ENDNEWType;
NEWTYPE dictionary /*Defines a dictionary with key dest_nam
Array(integer,item_type); /*Destination ProcessID*/
ENDNEWTYPE;
DCL dest_dic dictionary;
DCL number_of_entries integer := 0;
DCL dest_PId PId;
DCL name, dest_name charstring;
Idle
Streams to be written to a file or keyboard or ...
New destinations from Input-Manager
add_dest(name) /*from Input-Manager*/
value_to_dest (timestamp,obj_name,var_name,value,dest_name) /*from Chgd_Value_- or Ngets_Handler*/
Tests, if the Destination_Handler allready knows the dest.
exists:= (CALL DestExists (name))
exists:= (CALL DestExists (dest_name))
exists = True
exists = True
True
False
False
True
Creates a new process and adds it to a dictionary with key name
AddProcess (name)
dest_PId := (CALL getPId (dest_name))
Idle
write_value(timestamp, obj_name,var_name,value) TO dest_PId

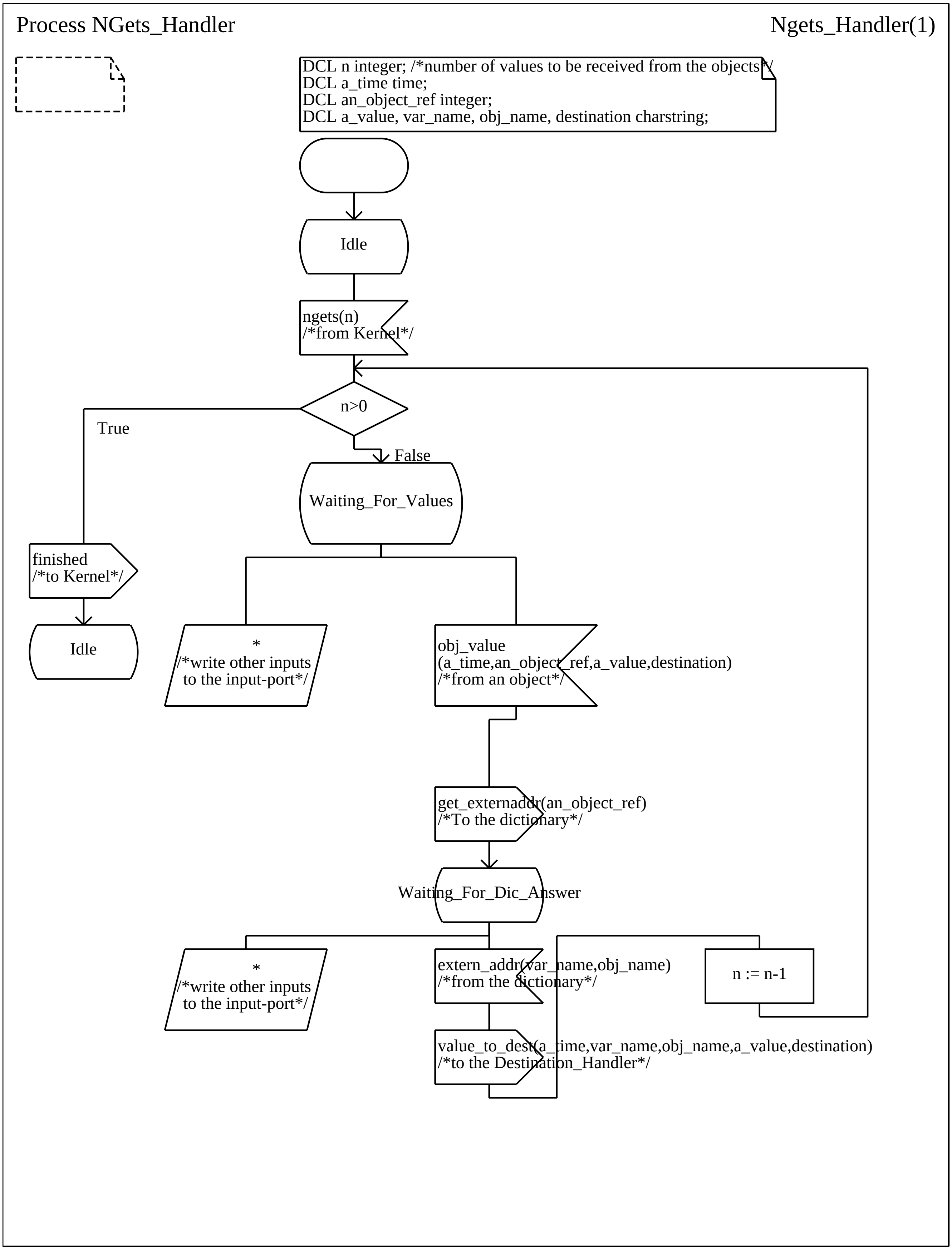
```
NEWTYPE item_type
STRUCT
   name charstring;
   Id PId;
ENDNEWType;
NEWTYPE dictionary             /*Defines a dictionary with key dest_name and value */
    Array(integer,item_type);    /*Destination ProcessID*/
ENDNEWTYPE;
DCL dest_dic dictionary;
DCL number_of_entries integer := 0;
DCL dest_PId PId;
DCL name, dest_name charstring;
DCL exists boolean;
DCL timestamp time;
DCL obj_name, var_name, value charstring;
```

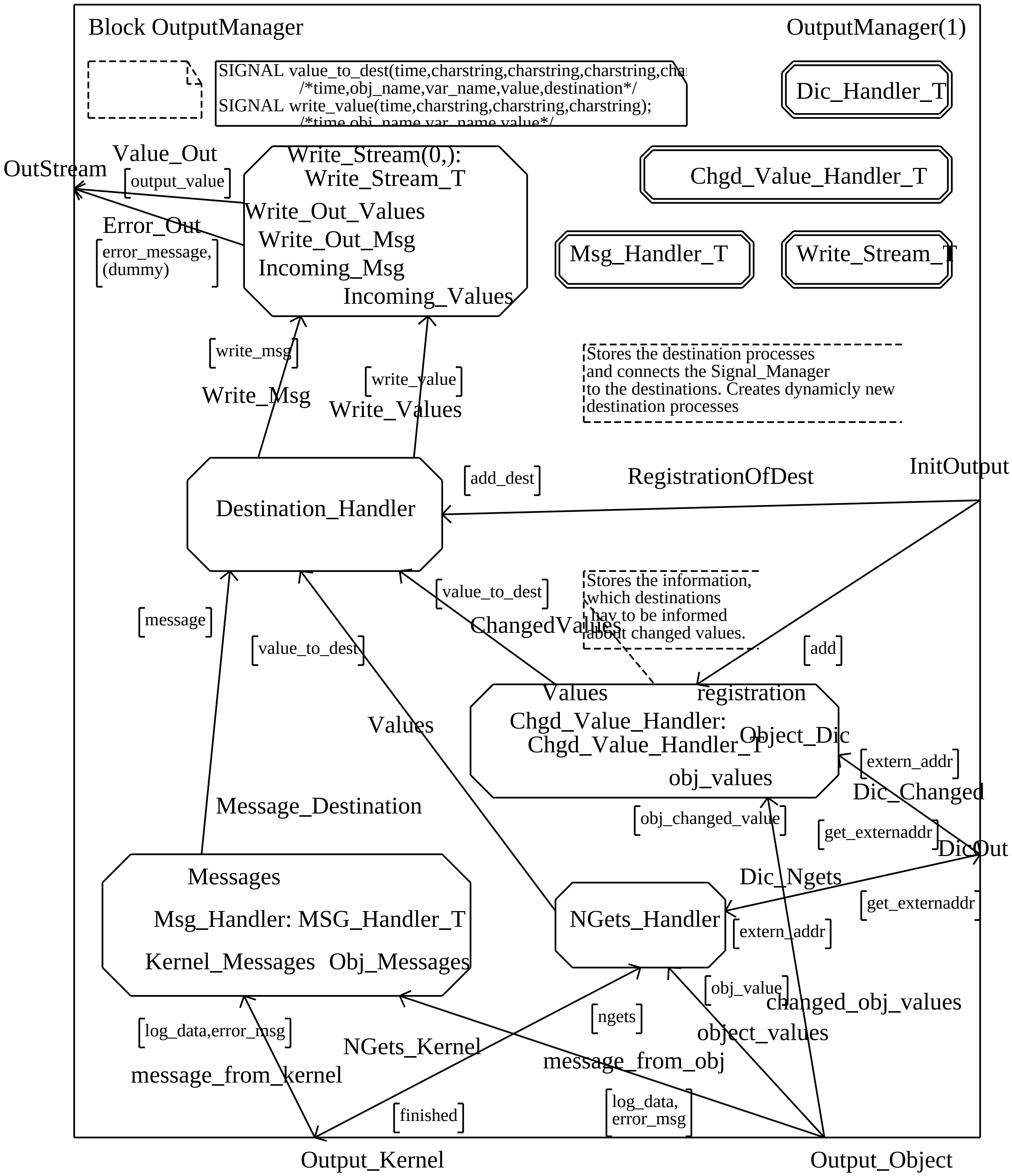

Procedure AddProcess
addprocess(1)
/* Needs variable dest_dictionary */
;FPAR IN dest_name charstring;
DCL end_of_dic integer;
Write_Stream
end_of_dic :=
number_of_entries+1
dest_dic(end_of_dic)!name:=
dest_name,
dest_dic(end_of_dic)!Id:=
OffSpring
Adds the process_ID
with key dest_name
to the dest_dictionary
number_of_entries :=
number_of_entries + 1



Procedure DestExists
exists(1)
;FPAR IN a_dest_name charstring;
RETURNS test boolean;
/*DCL a_value charstring;*/
DCL result,i, end_of_dic integer;
i := 0
end_of_dic := number_of_entries - i
False
Tests, if the end of
the agenda is reached.
If true: key not found.
If not: search goes on.
end_of_dic = 0
False
i := i + 1
True
StrCmp(dest_dic(i)!name,a_dest_name,result)
/*evtl muss noch TO PID */
test := False
result = 0
True
test :=True
key is in the dictionary
test

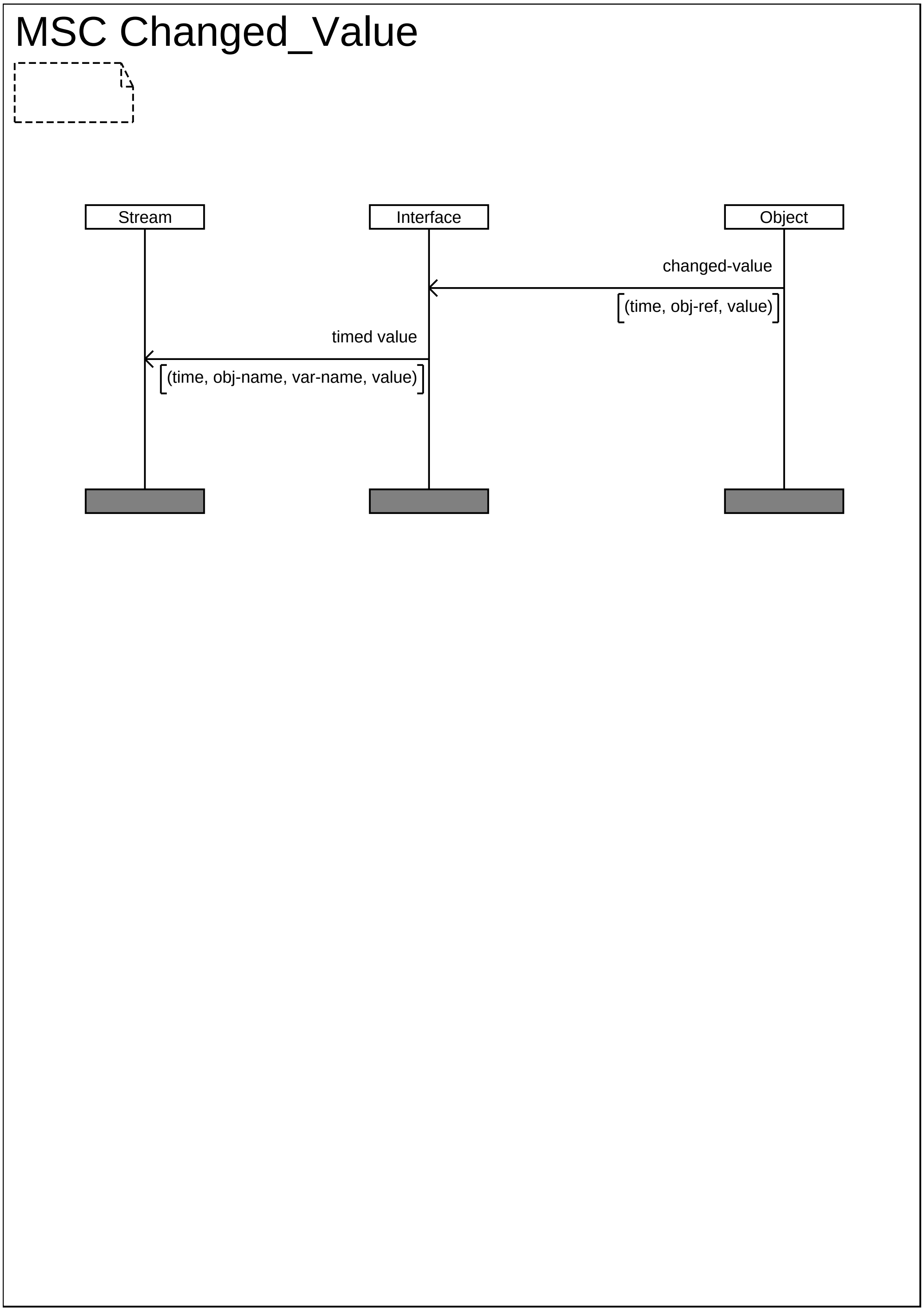

Procedure getPId
getprocessId(1)
;FPAR IN dest_name charstring;
RETURNS a_value PId;
/*DCL a_value charstring;*/
DCL result, end_of_dic, i integer;
i := 0
False
end_of_dic := number_of_entries - i
Tests, if the end of
the agenda is reached.
If true: key not found.
If not: search goes on.
end_of_dic = 0
False
i := i + 1
True
StrCmp(dest_dic(i)!name,dest_name,result)
/*evtl muss noch TO PID */
a_value := null
/*error???
key not found*/
result = 0
True
a_value :=
dest_dic(i)!Id
key is in the dictionary
a_value


Process NGets_Handler
Ngets_Handler(1)
DCL n integer; /*number of values to be received from the objects*/
DCL a_time time;
DCL an_object_ref integer;
DCL a_value, var_name, obj_name, destination charstring;
Idle
ngets(n)
/*from Kernel*/
n>0
True
False
Waiting_For_Values
finished
/*to Kernel*/
Idle
*
/*write other inputs
to the input-port*/
obj_value
(a_time,an_object_ref,a_value,destination)
/*from an object*/
get_externaddr(an_object_ref)
/*To the dictionary*/
Waiting_For_Dic_Answer
*
/*write other inputs
to the input-port*/
extern_addr(var_name,obj_name)
/*from the dictionary*/
n := n-1
value_to_dest(a_time,var_name,obj_name,a_value,destination)
/*to the Destination_Handler*/

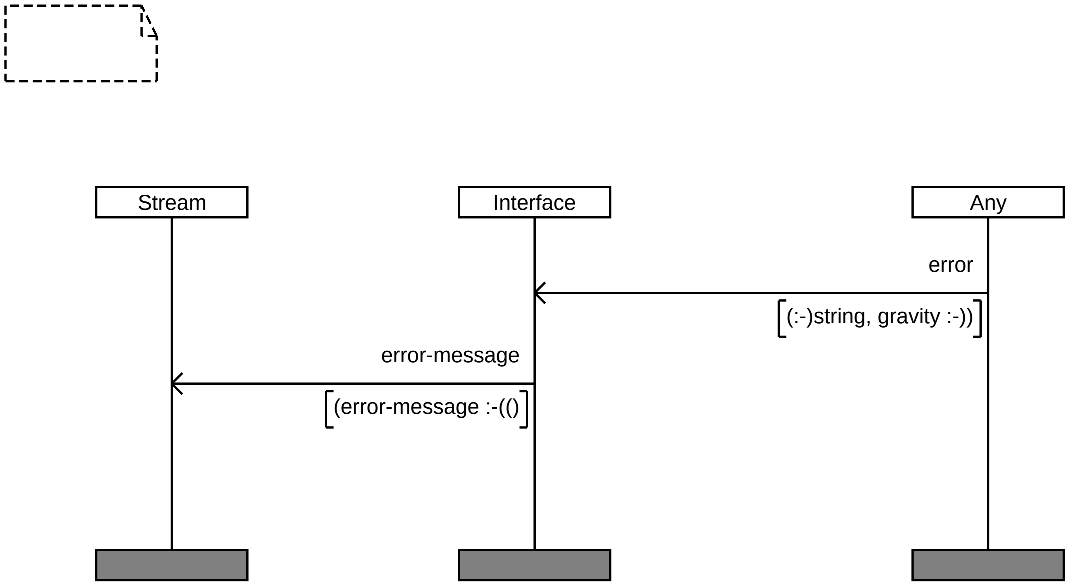


Block OutputManager
OutputManager(1)
SIGNAL value_to_dest(time,charstring,charstring,charstring,cha
/*time,obj_name,var_name,value,destination*/
SIGNAL write_value(time,charstring,charstring,charstring);
Dic_Handler_T
Chgd_Value_Handler_T
Msg_Handler_T
Write_Stream_T
Write_Stream(0,):
Write_Stream_T
Write_Out_Values
Write_Out_Msg
Incoming_Msg
Incoming_Values
OutStream
Value_Out
output_value
Error_Out
error_message,
(dummy)
write_msg
Write_Msg
write_value
Write_Values
Stores the destination processes
and connects the Signal_Manager
to the destinations. Creates dynamicly new
destination processes
Destination_Handler
add_dest
RegistrationOfDest
InitOutput
message
value_to_dest
value_to_dest
ChangedValues
Stores the information,
which destinations
hav to be informed
about changed values.
add
Values
registration
Chgd_Value_Handler:
Chgd_Value_Handler_T
Object_Dic
obj_values
Values
extern_addr
Dic_Changed
Message_Destination
obj_changed_value
get_externaddr
DicOut
Dic_Ngets
get_externaddr
Messages
Msg_Handler: MSG_Handler_T
Kernel_Messages
Obj_Messages
NGets_Handler
extern_addr
obj_value
changed_obj_values
ngets
object_values
log_data,error_msg
NGets_Kernel
message_from_obj
message_from_kernel
finished
log_data,
error_msg
Output_Kernel
Output_Object

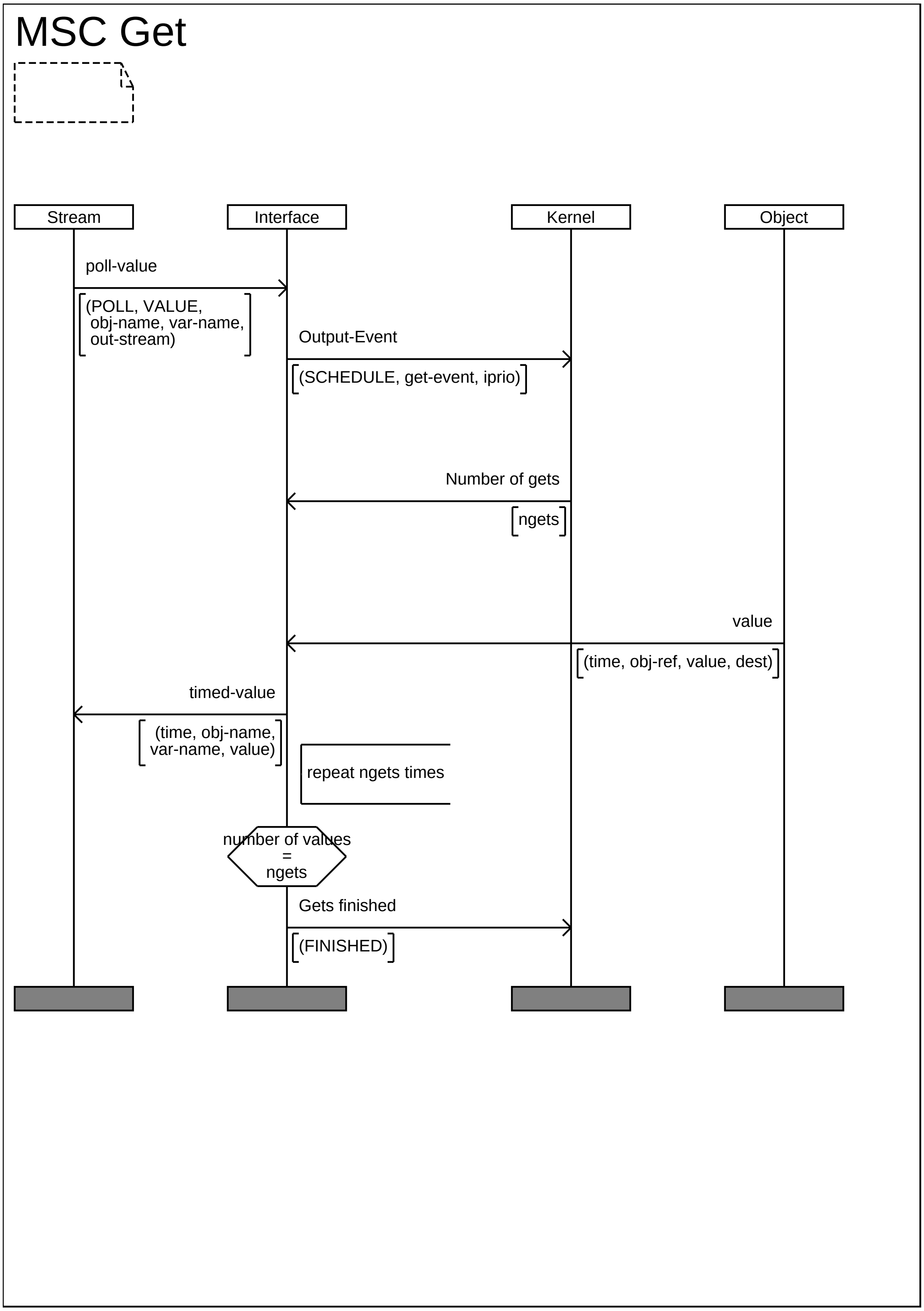
```
SIGNAL value_to_dest(time,charstring,charstring,charstring,charstring);
          /*time,obj_name,var_name,value,destination*/
SIGNAL write_value(time,charstring,charstring,charstring);
          /*time,obj_name,var_name,value*/
SIGNAL write_msg(charstring);
SIGNAL message(charstring);
SIGNALLIST dummy = result, ack, log;
```

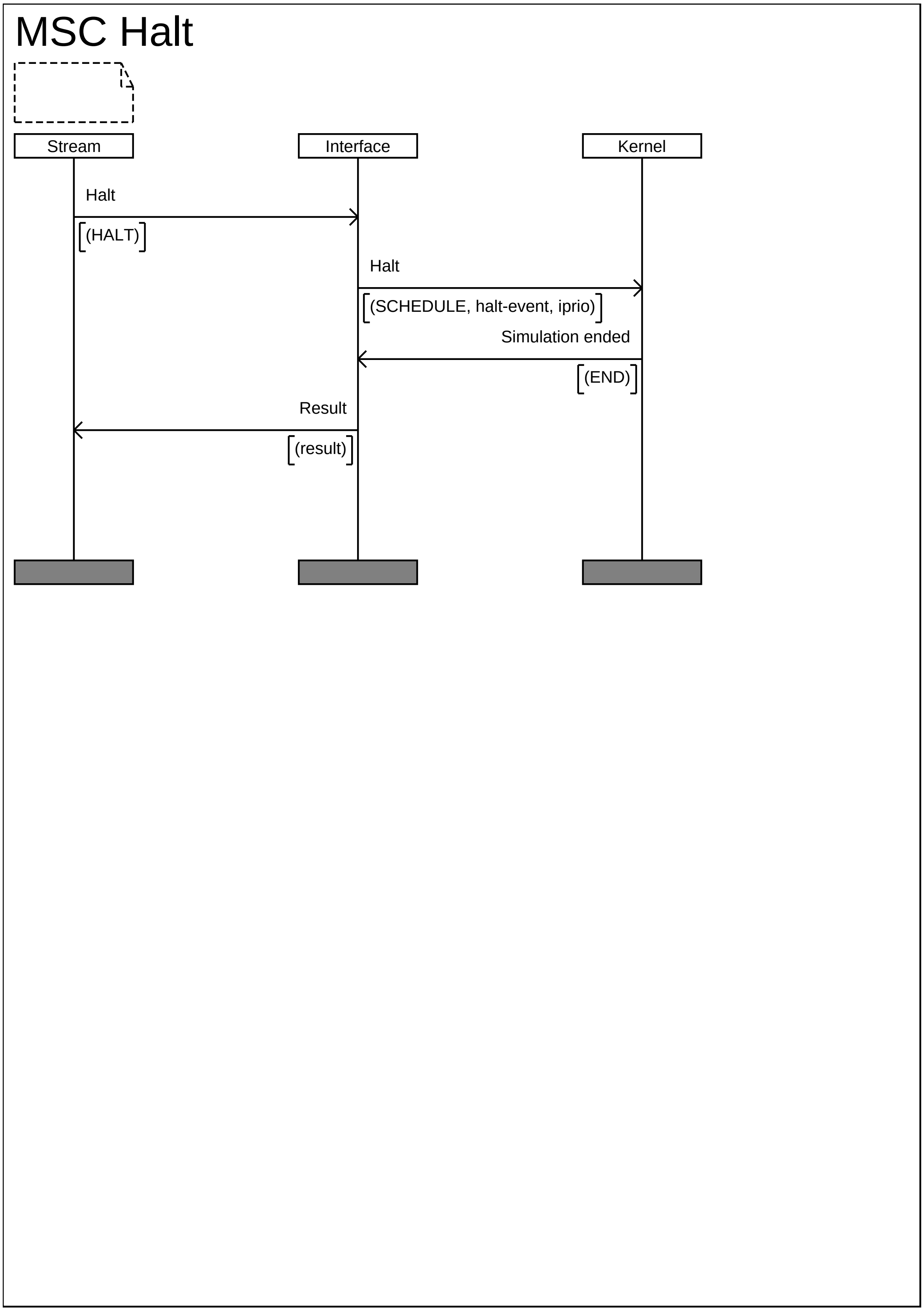

MSC Changed_Value
Stream
Interface
Object
changed-value
(time, obj-ref, value)
timed value
(time, obj-name, var-name, value)

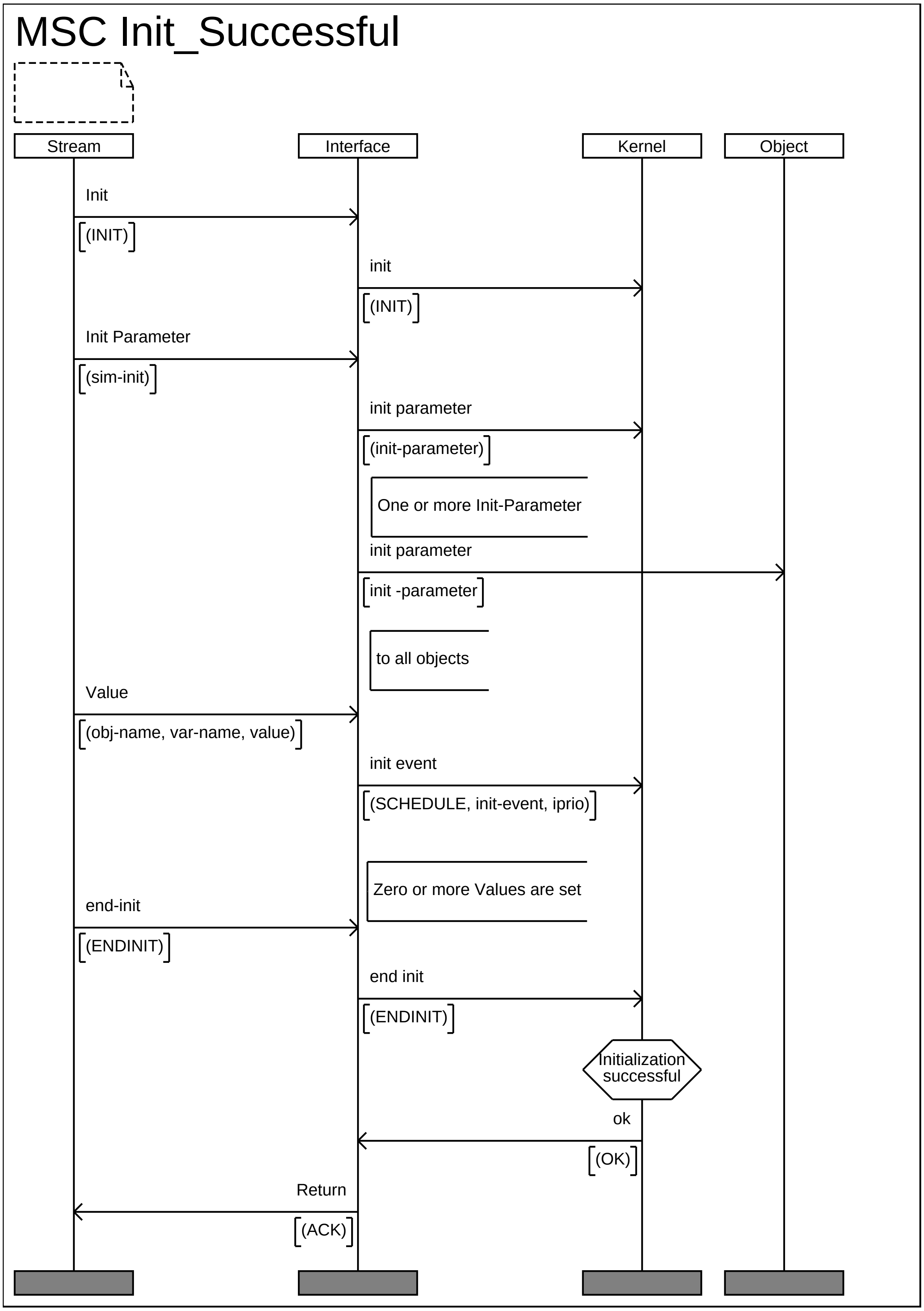
MSC Error



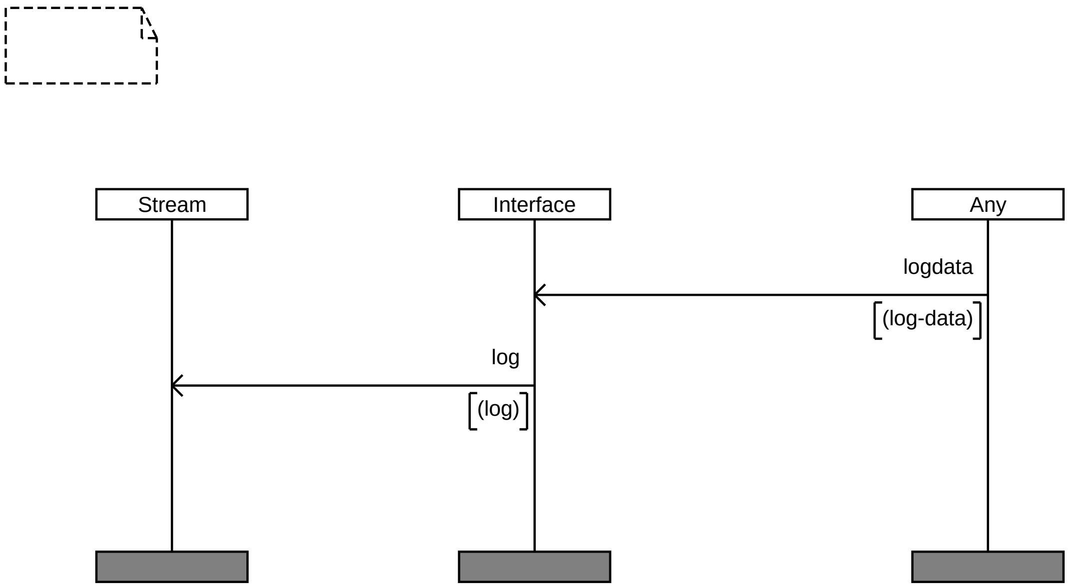
Stream
Interface
Any
error
(:-)string, gravity :-))
error-message
(error-message :-(()

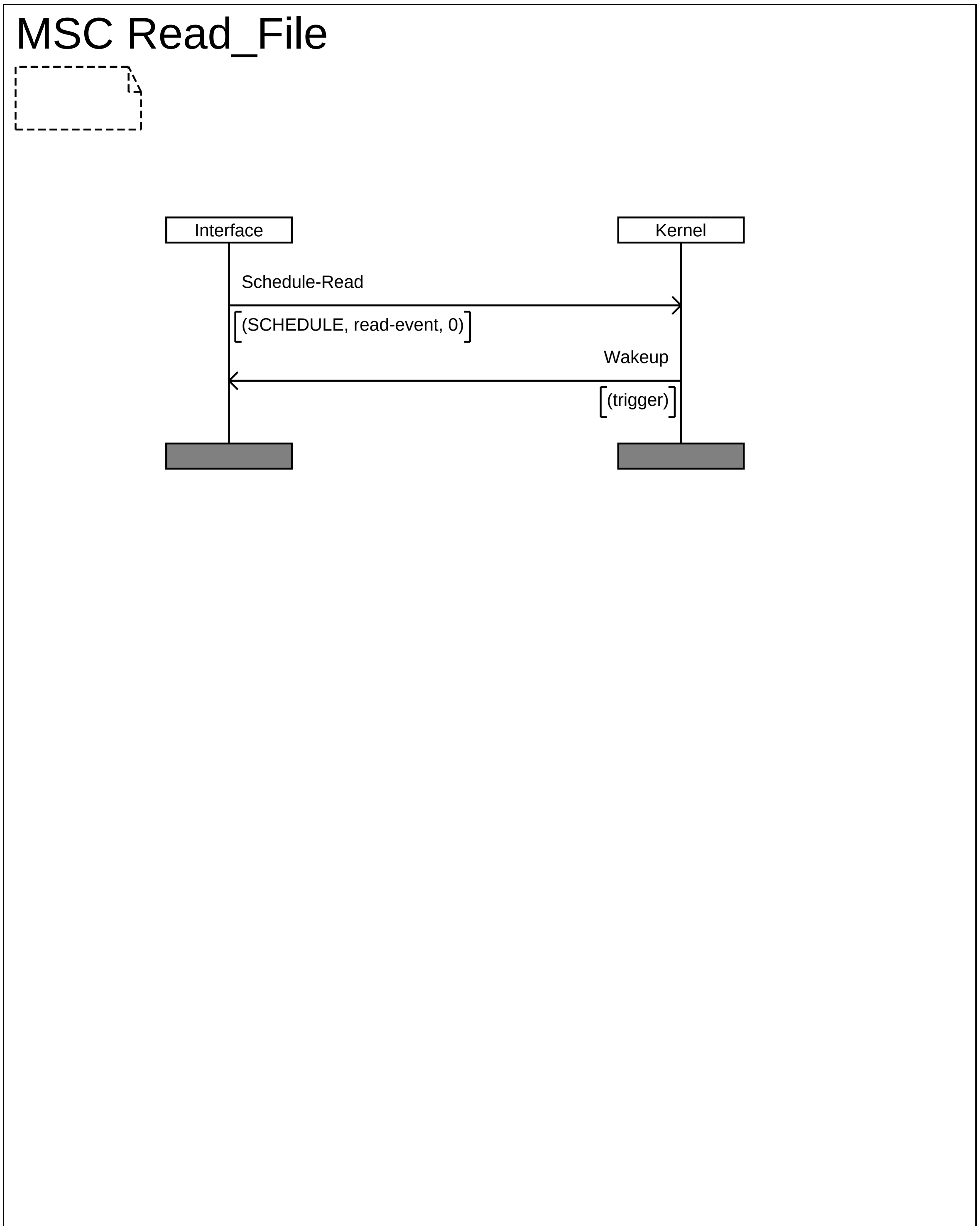

MSC Get
Stream
Interface
Kernel
Object
poll-value
(POLL, VALUE,
obj-name, var-name,
out-stream)
Output-Event
(SCHEDULE, get-event, iprio)
Number of gets
ngets
value
(time, obj-ref, value, dest)
timed-value
(time, obj-name,
var-name, value)
repeat ngets times
number of values
=
ngets
Gets finished
(FINISHED)


MSC Halt
Stream
Interface
Kernel
Halt
(HALT)
Halt
(SCHEDULE, halt-event, iprio)
Simulation ended
(END)
Result
(result)

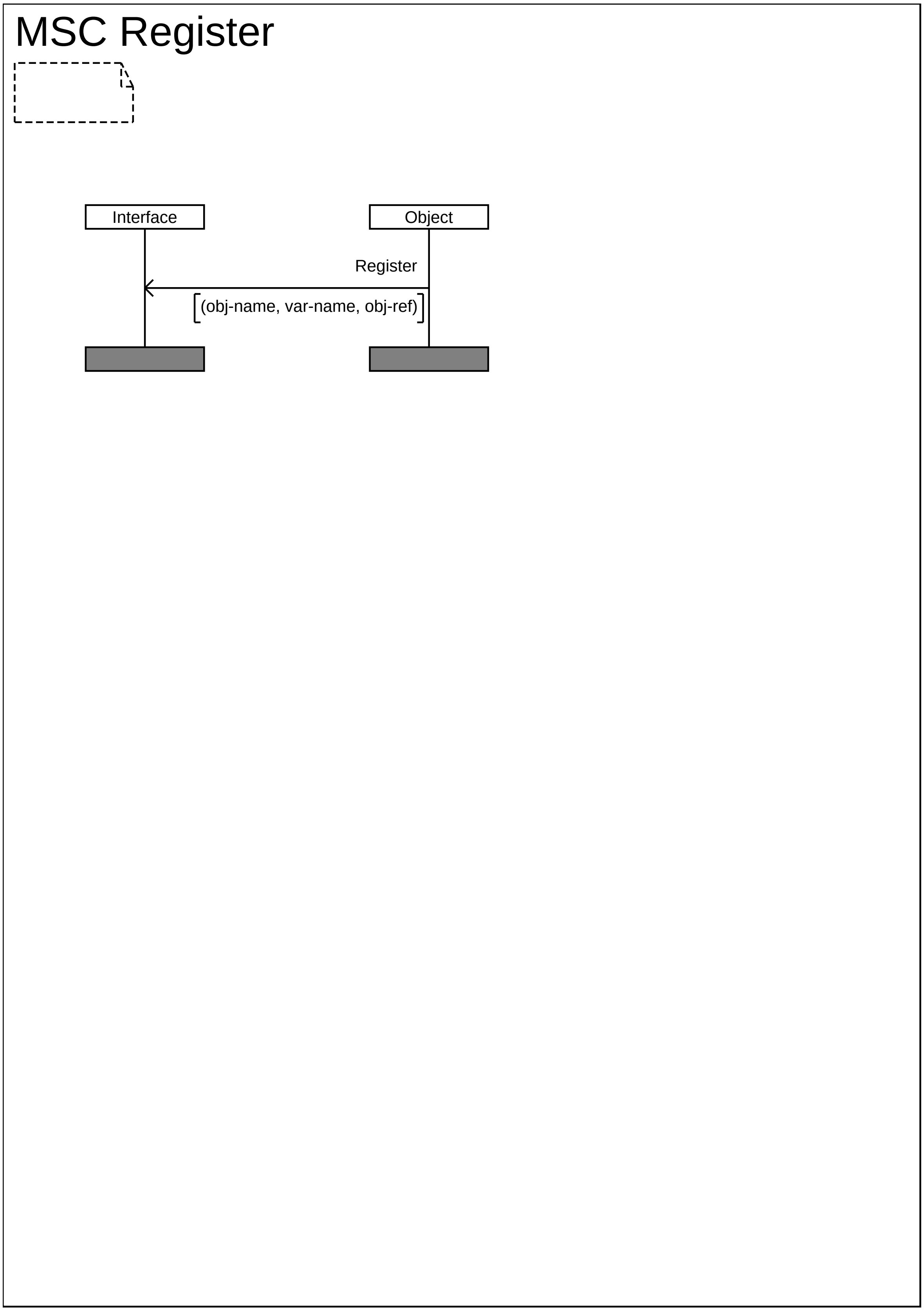

MSC Init_Successful
Stream
Interface
Kernel
Object
Init
(INIT)
init
(INIT)
Init Parameter
(sim-init)
init parameter
(init-parameter)
One or more Init-Parameter
init parameter
init -parameter
to all objects
Value
(obj-name, var-name, value)
init event
(SCHEDULE, init-event, iprio)
Zero or more Values are set
end-init
(ENDINIT)
end init
(ENDINIT)
Initialization successful
ok
(OK)
Return
(ACK)

MSC Log

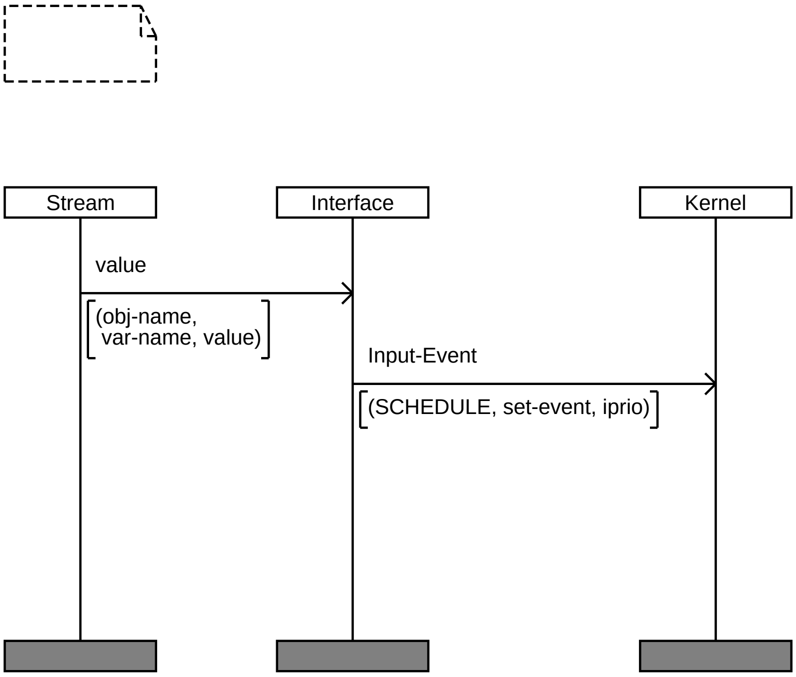


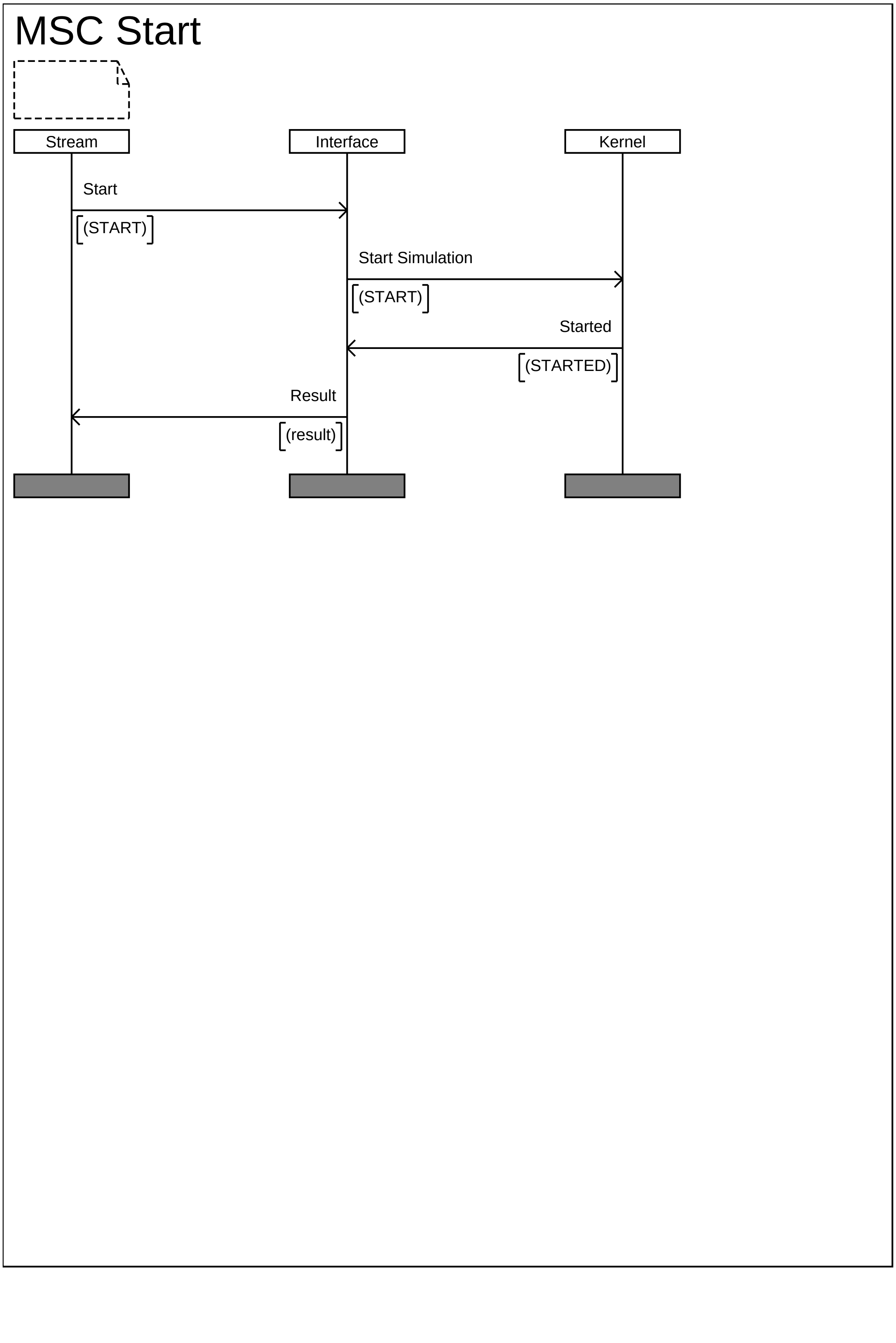


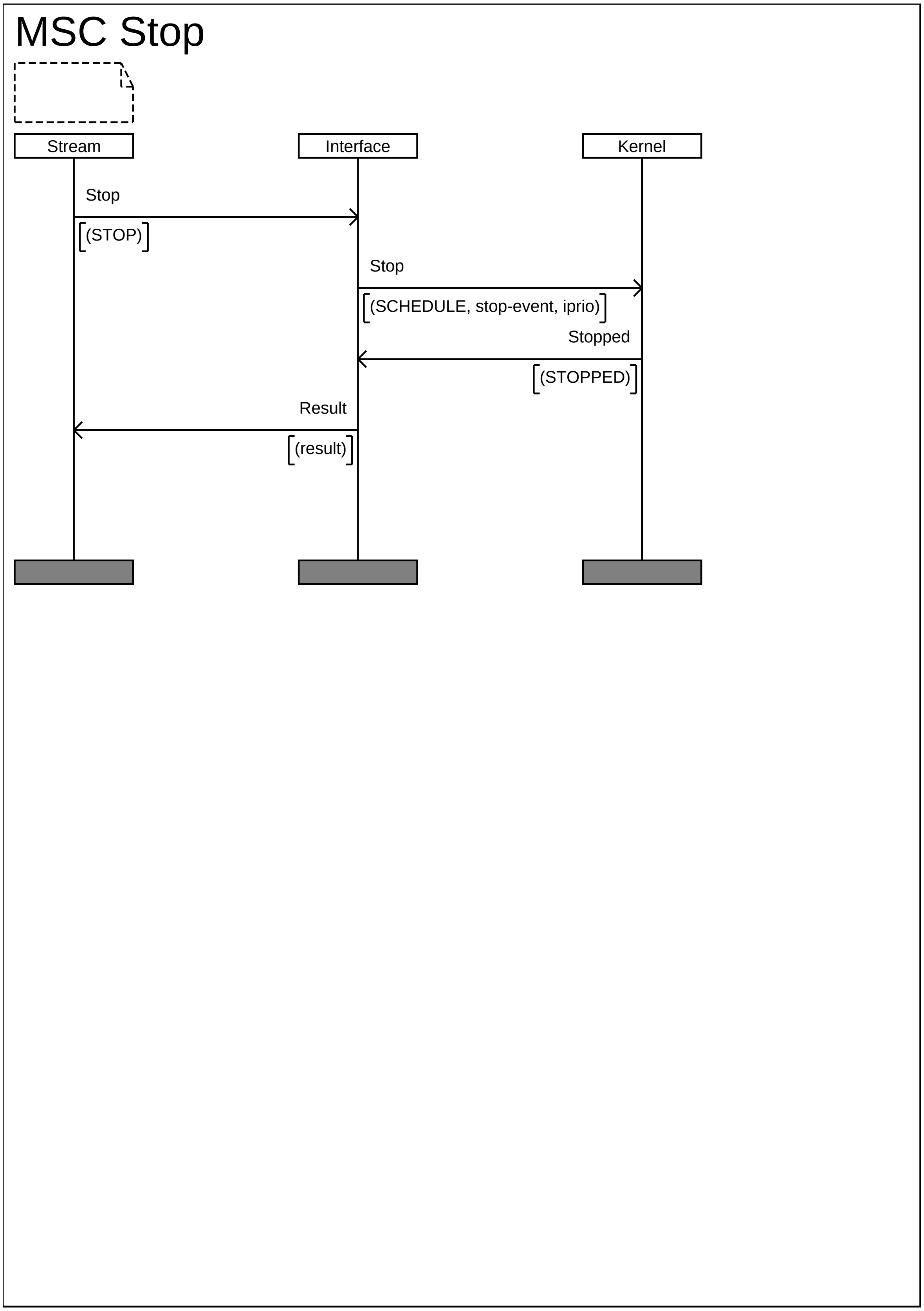
MSC Read_File
Interface
Kernel
Schedule-Read
(SCHEDULE, read-event, 0)
Wakeup
(trigger)


MSC Register
Interface
Object
Register
(obj-name, var-name, obj-ref)

MSC Set


Stream
Interface
Kernel
value
(obj-name, var-name, value)
Input-Event
(SCHEDULE, set-event, iprio)


MSC Start
Stream
Interface
Kernel
Start
(START)
Start Simulation
(START)
Started
(STARTED)
Result
(result)


MSC Stop
Stream
Interface
Kernel
Stop
(STOP)
Stop
(SCHEDULE, stop-event, iprio)
Stopped
(STOPPED)
Result
(result)