Table of Contents

Diagram Structure

- ExternalSignals
- FilingSystem
  - FileSystems
    - Filing
      - CloseFile
      - OpenFile
      - ReadInt
      - ReadReal
      - ReadString
      - ReadValue
      - WriteInt
      - WriteReal
      - WriteString
      - WriteValue
- GlobalDataTypes
- InternalSignals
- Simulator
  - Controller
    - Control
      - DumpAnything
      - DumpValues
      - GetID
      - RetrieveDump
  - Interrface
    - Interfacing
      - DumpAnything
      - Insert
      - InsertObject
        - InsertList
      - RetrieveDump
  - Kernel
    - Clock
      - DumpAnything
      - DumpValues

        - RetrieveDump
    - EventCollection
        - DumpAnything
        - DumpValues
        - RetrieveDump
    - EventHandling
    - PassThru
        - DumpAnything
        - DumpValues
        - RetrieveDump
- x:y FileSystem : FileSystems

Associated Documents

```
USE GlobalDataTypes;
USE FilingSystem;
```

Package ExternalSignals


```
/* ===================================================== */
/* = I. Signal definitions for SimulationObject communication 1(3)= */
/* ===================================================== */
```

```
SIGNAL /* Get1 group */
  get_visible_variable(Visible_variable_IDs, Event_stream_IDs, Timestamps),
  get_values(Timestamps),
  get_anything(Timestamps);

SIGNALLIST
  Get1 =
    get_visible_variable, get_values, get_anything;
```

```
/* Get1 group */
```

```
SIGNAL /* Set1 group */
  set_parameter(Parameter_IDs, Values),
  set_visible_variable(Visible_variable_IDs, Values, Timestamps8),
  set_independent_variable(Independent_variable_IDs,
    Values, Timestamps8),
  add_output_configuration(Visible_variable_IDs,
    Event_stream_IDs, Timestamps8),
  empty_output_configuration(Visible_variable_IDs, Timestamps8),
  set_internal_variable(Internal_variable_IDs, Values, Timestamps8),
  set_max_period(Durations, Timestamps8);

SIGNALLIST
  Set1 =
    set_parameter, set_visible_variable, set_independent_variable,
    add_output_configuration, empty_output_configuration,
    set_internal_variable, set_max_period;
```

```
/* Set1 group */
```

```
USE GlobalDataTypes;
USE FilingSystem;
```

Package ExternalSignals


```
/* ========================================================== */
/* = I. Signal definitions for SimulationObject communication 2(3) = */
/* ========================================================== */
```

```
SIGNAL /* Value group */
  parameter(Parameter_IDs, Values),
  visible_variable(Visible_variable_IDs, Values, Event_stream_IDs),
  output_configuration(Visible_variable_IDs,
    LIST_OF_Event_stream_IDs),
  internal_variable(Internal_variable_IDs, Values),
  max_period(Object_IDs, Durations),
  value_changed(Visible_variable_IDs, Values,
    LIST_OF_Event_stream_IDs);

SIGNALLIST
  Value =
    parameter, visible_variable, output_configuration,
    internal_variable, max_period, value_changed;
```

/* Value group */

```
SIGNAL
  simulate (Timestamps);
```

/* simulate */

```
SIGNAL /* Message group */
  object_finished,
  object_warning(Strings),
  object_error(Strings);

SIGNALLIST
  Message =
    object_finished, object_warning, object_error;
```

/* Message group */

```
SIGNAL /* Schedule group */
  schedule_sim_event(Timestamps, Object_IDs, Durations),
  reschedule_event(Event_IDs, Durations),
  unschedule_event(Event_IDs);

SIGNALLIST
  ScheduleSO =
    schedule_sim_event, reschedule_event, unschedule_event;
```

/* Schedule group */

```
USE GlobalDataTypes;
USE FilingSystem;
```

Package ExternalSignals


```
/* ====================================================== */
/* = I. Signal definitions for SimulationObject communication 3(3) = */
/* ====================================================== */
```

```
SIGNAL /* AdminToSO group */
  initialize,
  get_ready,
  scheduled_event(Timestamps, Object_IDs, Values, Durations, Event_IDs),
  schedule_error(Timestamps, Visible_Variable_IDs, Values, Durations, Strings),
  id(IDs, Names);

SIGNALLIST
  AdminToSO =
    initialize, get_ready, scheduled_event,
    schedule_error, id;
```

```
/* AdminToSO group */
```

```
SIGNAL /* AdminFromSO group */
  register_object(Object_names,
    LIST_OF_Parameter_names_Parameter_IDs,
    LIST_OF_Visible_variable_names_Visible_variable_IDs,
    LIST_OF_Internal_variable_names_Internal_variable_IDs,
    Durations),
  get_id(Names);

SIGNALLIST
  AdminFromSO =
    register_object, get_id;
```

```
/* AdminFromSO group */
```

```
SIGNAL /* Get' group */
  get_max_period(Timestamps),
  get_parameter(Parameter_IDs);

SIGNALLIST /* Get' group */
  Get2 =
    get_parameter, get_visible_variable, get_max_period;
```

```
/* Get2 group */
```

```
/* For communication among simulation objects only. */
/* These do not belong to the system. */
```

```
SIGNALLIST /* Value' group */
  Value2 =
    parameter, visible_variable, max_period;
```

```
/* Value2 group */
```

```
USE GlobalDataTypes;
USE FilingSystem;
```

Package ExternalSignals


```
/* ================================================== */
/* = II. Signal definitions for Event Stream Communication 1(1) = */
/* ================================================== */
```

```
SIGNAL /* Input group */
  io(TimestampsStar, Object_names, Messages),
  io_period(TimestampsStar, Object_names, Messages, Durations);

SIGNALLIST
  Inputs =
    io, io_period;
```

```
/* Input group */
```

```
SIGNAL /* AdminFromES group */
  register_event_stream(Event_stream_names),
  get_simulation_time,
  event_stream_warning(Strings),
  event_stream_error(Strings);

SIGNALLIST
  AdminFromES =
    register_event_stream, get_simulation_time,
    get_simulation_time, event_stream_warning,
    event_stream_error;
```

```
/* AdminFromES group */
```

```
SIGNALLIST
  Outputs =
    io;
```

```
/* Output group */
```

```
SIGNAL /* AdminToES group */
  simulation_time(Timestamps),
  io_error(Timestamps, Object_names, Messages,
    Durations, Strings),
  finished;

SIGNALLIST
  AdminToES =
    initialize, simulation_time,
    io_error, finished;
```

```
/* AdminToES group */
```

USE GlobalDataTypes;
USE FilingSystem;


```
/* ===================================================== */
/* = III. Signal definitions for Control Stream communication 1(2) = */
/* ===================================================== */
```

```
SIGNAL /* Initialize_simulator group */
  start_initializing,
  alter_parameter(Parameter_names, Values),
  set_time_mode_fast,
  set_time_mode(RPlus, Durations, Boolean),
  set_simulation_start(Timestamps),
  retrieve_dump(File_names);

SIGNALLIST
  Initialize_simulator =
    start_initializing, alter_parameter, set_time_mode_fast,
    set_time_mode, set_simulation_start, retrieve_dump,
    get_ready;
```

/* Initialize_simulator group */

```
SIGNAL /* Control_simulator group */
  alter_independent_variable(Independent_variable_names,Values),
  set_simulation_end(TimestampsStar),
  unset_simulation_end,
  set_halting_point(TimestampsStar),
  unset_halting_point(TimestampsStar),
  stop_simulation,
  resume_simulation,
  dump_values(File_names),
  dump_anything(File_names);

SIGNALLIST
  Control_simulator =
    alter_independent_variable,
    set_simulation_end, unset_simulation_end,
    set_halting_point, unset_halting_point,
    stop_simulation, resume_simulation,
    dump_values, dump_anything;
```

/* Control_simulator group */

```
USE GlobalDataTypes;
USE FilingSystem;
```

Package ExternalSignals


```
/* ======================================================== */
/* = III. Signal definitions for Control Stream communication 2(2) = */
/* ======================================================== */
```

```
SIGNAL /* Initialize_return group */
  initialize_OK,
  initialize_error(Strings),
  modify_warning(Strings),
  get_ready_OK,
  get_ready_error(Strings);

SIGNALLIST
  Initialize_return =
    initialize_OK, initialize_error,
    modify_warning, get_ready_OK, get_ready_error;
```

/* Initialize_return group */

```
SIGNAL /* Simulator_return group */
  simulation_stopped,
  simulation_OK,
  simulation_slacked,
  simulation_warning(Strings),
  simulation_error(Strings);

SIGNALLIST
  Simulator_return =
    simulation_stopped, simulation_OK,
    simulation_slacked, simulation_warning,
    simulation_error;
```

/* Simulator_return group */

```
USE GlobalDataTypes;
```

Package FilingSystem

1(1)

```
/* Data types to be exported */

SYNTYPE File_names                    /* the set of allowed names for files; depends on the file system */
  = Names
ENDSYNTYPE;

SYNTYPE File_IDs                      /* the set of corresponding internal references */
  = Integer CONSTANTS >= 0
ENDSYNTYPE;

SYNONYM f_error
  File_IDs = 0;

NEWTYPE File_modes
  LITERALS
    f_read, f_write;
ENDNEWTYPE File_modes;
```

```
/* Open / close procedures to be exported */

REMOTE PROCEDURE OpenFile NODELAY;
FPAR IN File_names, File_modes;
RETURNS File_IDs;

REMOTE PROCEDURE CloseFile NODELAY;
FPAR IN File_IDs;
```

FileSystems

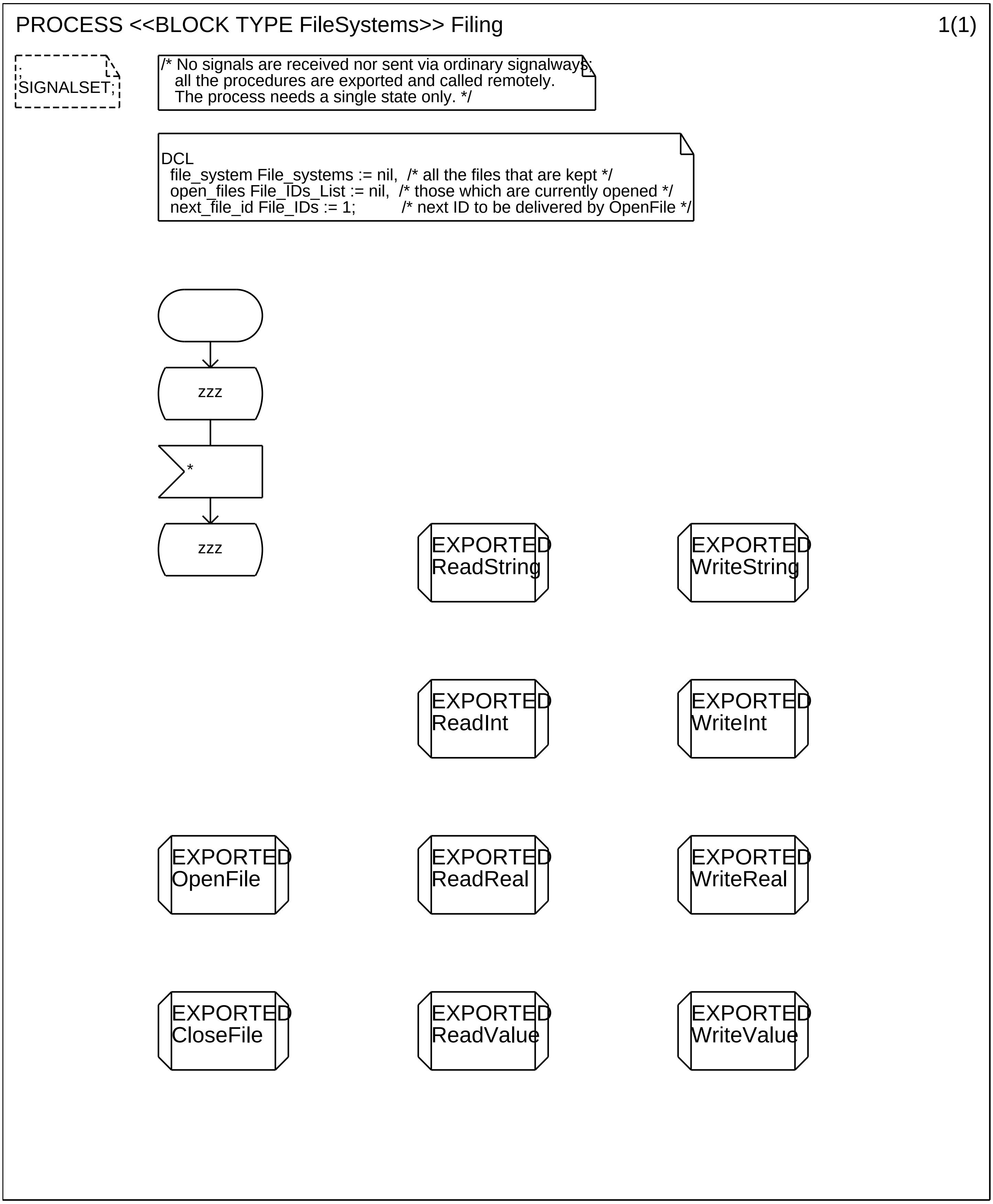
```
/* Read procedures to be exported */

REMOTE PROCEDURE ReadString NODELAY;
FPAR IN File_IDs;
RETURNS Charstring;

REMOTE PROCEDURE ReadInt NODELAY;
FPAR IN File_IDs;
RETURNS Integer;

REMOTE PROCEDURE ReadReal NODELAY;
FPAR IN File_IDs;
RETURNS Real;

REMOTE PROCEDURE ReadValue NODELAY;
FPAR IN File_IDs;
RETURNS Values;
```

```
/* Write procedures to be exported */

REMOTE PROCEDURE WriteString NODELAY;
FPAR IN File_IDs, Charstring;

REMOTE PROCEDURE WriteInt NODELAY;
FPAR IN File_IDs, Integer;

REMOTE PROCEDURE WriteReal NODELAY;
FPAR IN File_IDs, Real;

REMOTE PROCEDURE WriteValue NODELAY;
FPAR IN File_IDs, Values;
```

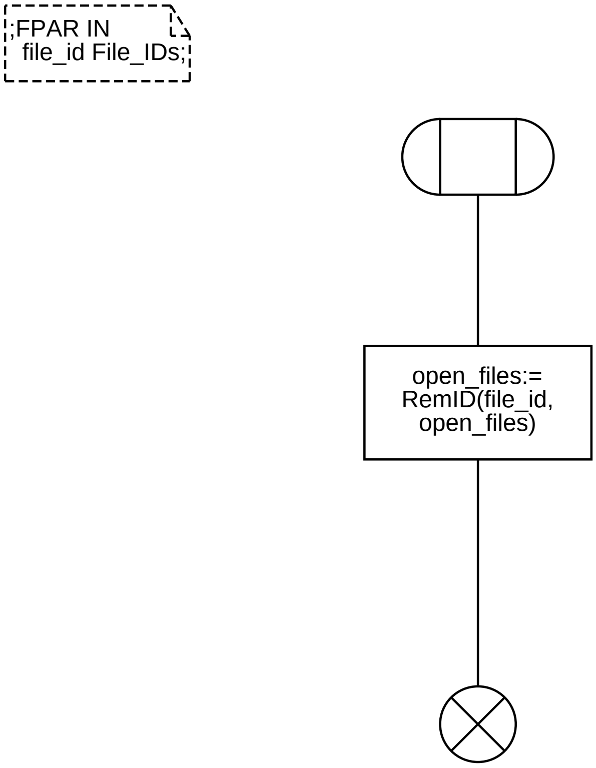
BLOCK TYPE FileSystems

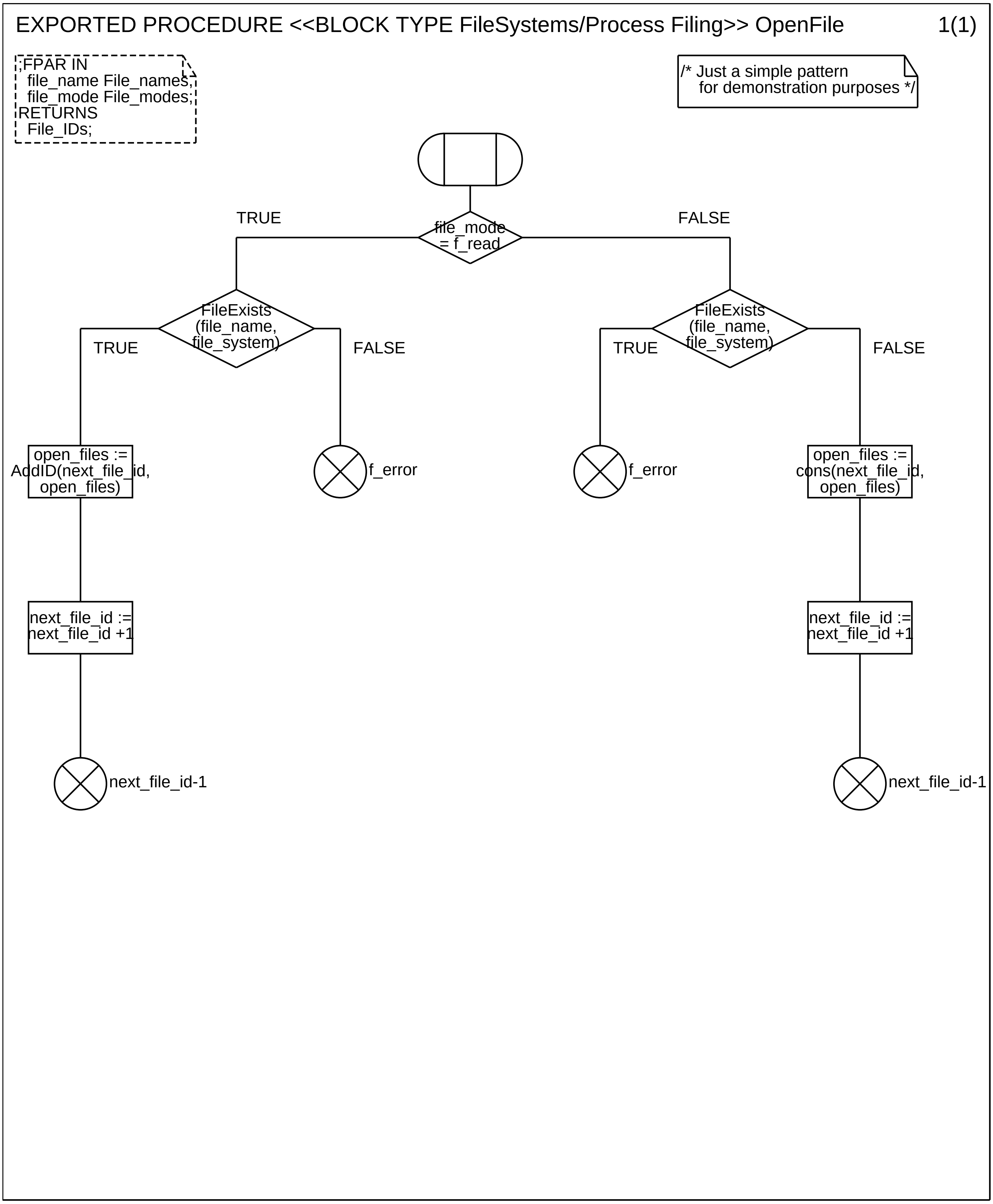

```
NEWTYPE Files
  STRUCT
    file_name File_names;
    file_contents Charstring;
ENDNEWTYPE Files;

NEWTYPE File_systems
  List(Files)
ADDING
OPERATORS
  FileExists: File_names, File_Systems  -> BOOLEAN
AXIOMS
  FOR ALL n IN File_names (
    FOR ALL f IN Files (
      FOR ALL fs IN File_systems (
        FileExists(n,nil) == FALSE;
        FileExists(n,cons(f,fs)) ==
          n=file_nameEXTRACT!(f) OR FileExists(n,fs);
      )
    )
  )
ENDNEWTYPE File_systems;


NEWTYPE File_IDs_List
  List(File_IDs)
ADDING
OPERATORS
  AddID: File_IDs, File_IDs_List -> File_IDs_List;
  RemID: File_IDs, File_IDs_List -> File_IDs_List;
AXIOMS
  FOR ALL l IN File_IDs_List (
    FOR ALL i,j IN File_IDs (
      AddID(i,l) == cons(i,l);
      RemID(i,nil) == nil;
      RemID(i,cons(j,l)) ==
        IF i=j THEN l
        ELSE RemID(i,l) FI;
    )
  )
ENDNEWTYPE File_IDs_List;
```

Filing

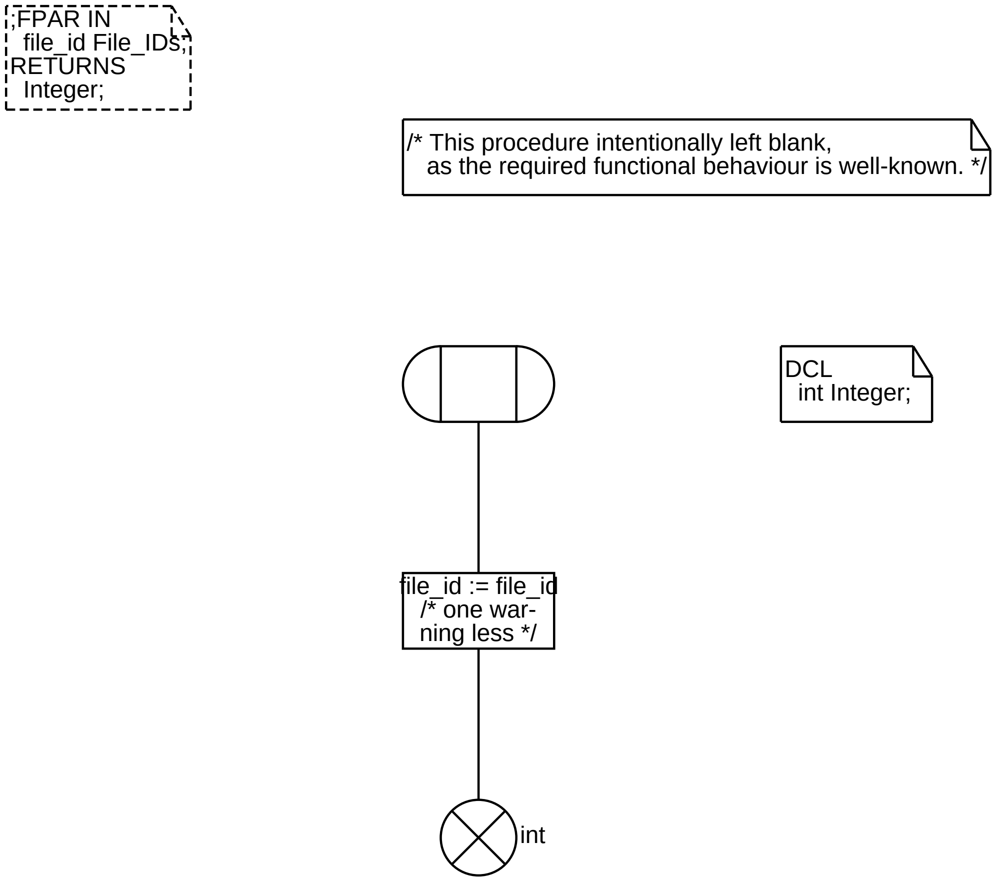

PROCESS <<BLOCK TYPE FileSystems>> Filing
1(1)
SIGNALSET;
/* No signals are received nor sent via ordinary signalways;
all the procedures are exported and called remotely.
The process needs a single state only. */
DCL
file_system File_systems := nil, /* all the files that are kept */
open_files File_IDs_List := nil, /* those which are currently opened */
next_file_id File_IDs := 1; /* next ID to be delivered by OpenFile */
zzz
*
zzz
EXPORTED ReadString
EXPORTED WriteString
EXPORTED ReadInt
EXPORTED WriteInt
EXPORTED OpenFile
EXPORTED ReadReal
EXPORTED WriteReal
EXPORTED CloseFile
EXPORTED ReadValue
EXPORTED WriteValue

EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> CloseFile

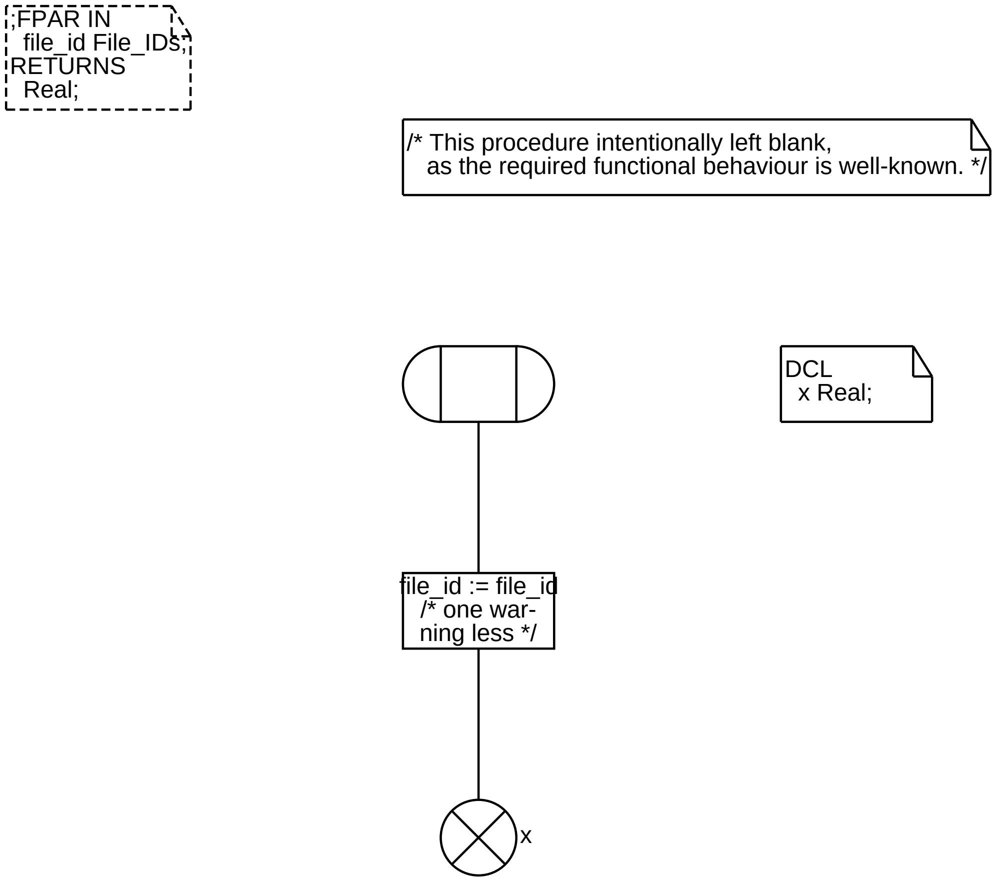


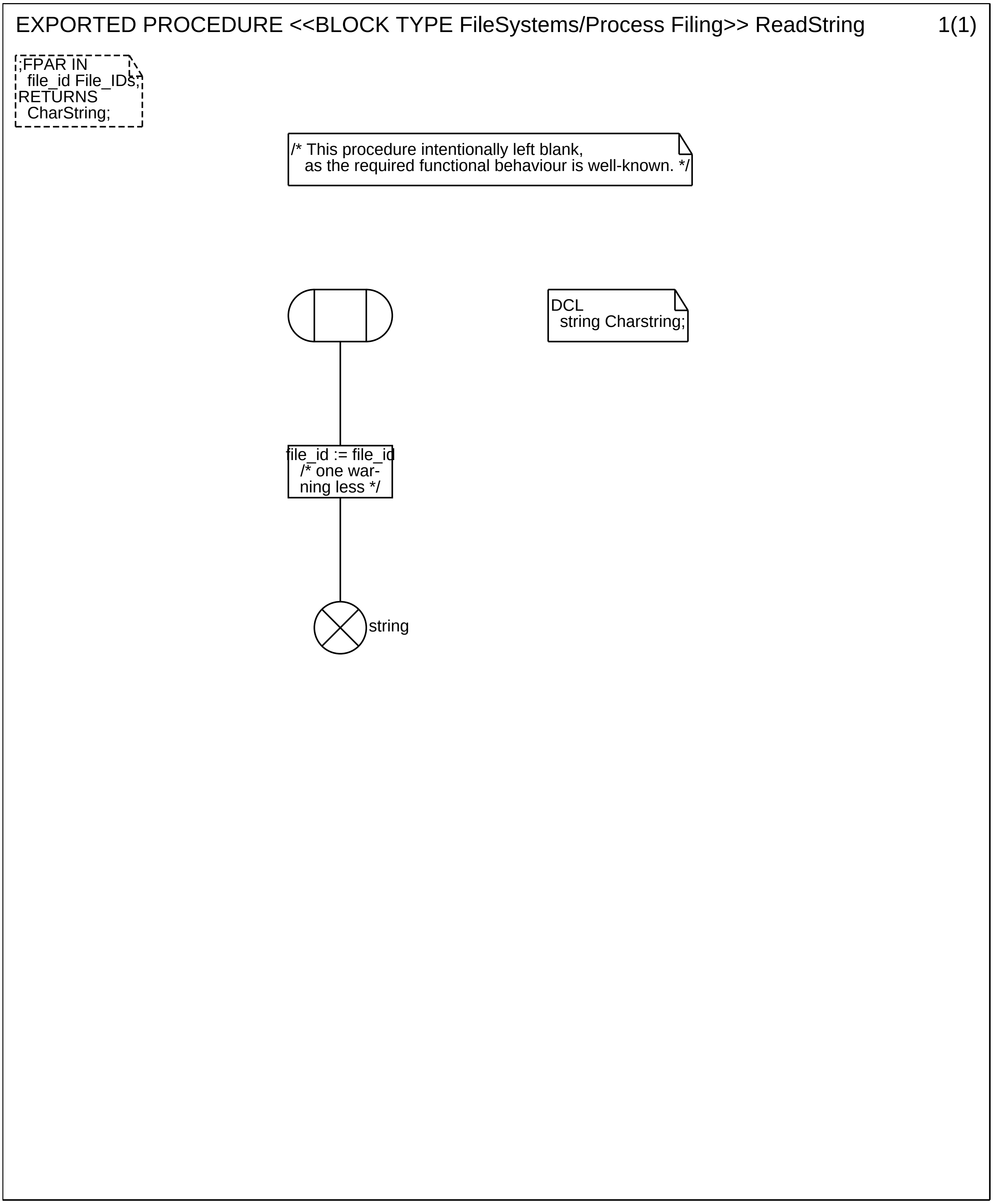

EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> OpenFile
1(1)
;FPAR IN
file_name File_names;
file_mode File_modes;
RETURNS
File_IDs;
/* Just a simple pattern
for demonstration purposes */
TRUE
file_mode = f_read
FALSE
FileExists (file_name, file_system)
TRUE
FALSE
FileExists (file_name, file_system)
TRUE
FALSE
open_files := AddID(next_file_id, open_files)
f_error
f_error
open_files := cons(next_file_id, open_files)
next_file_id := next_file_id +1
next_file_id := next_file_id +1
next_file_id-1
next_file_id-1

EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> ReadInt 1(1)

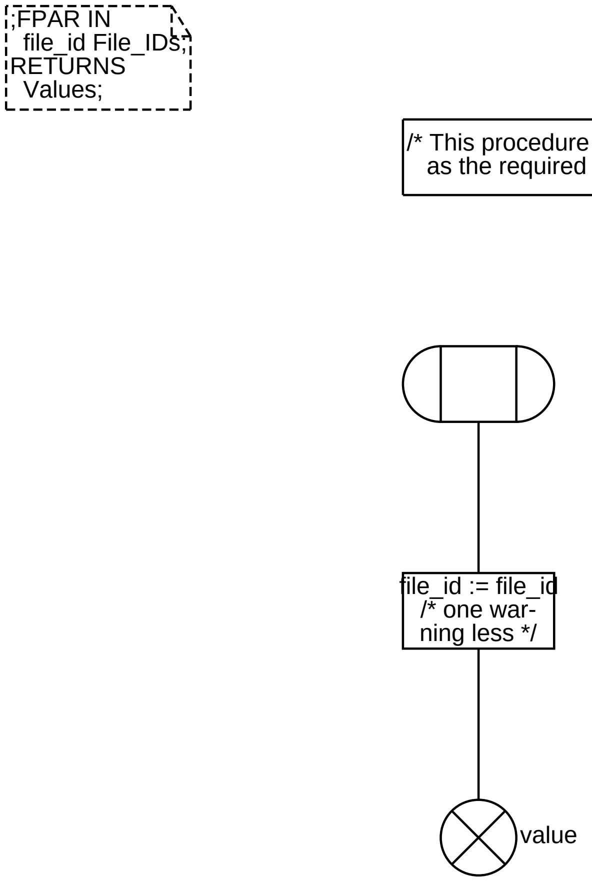


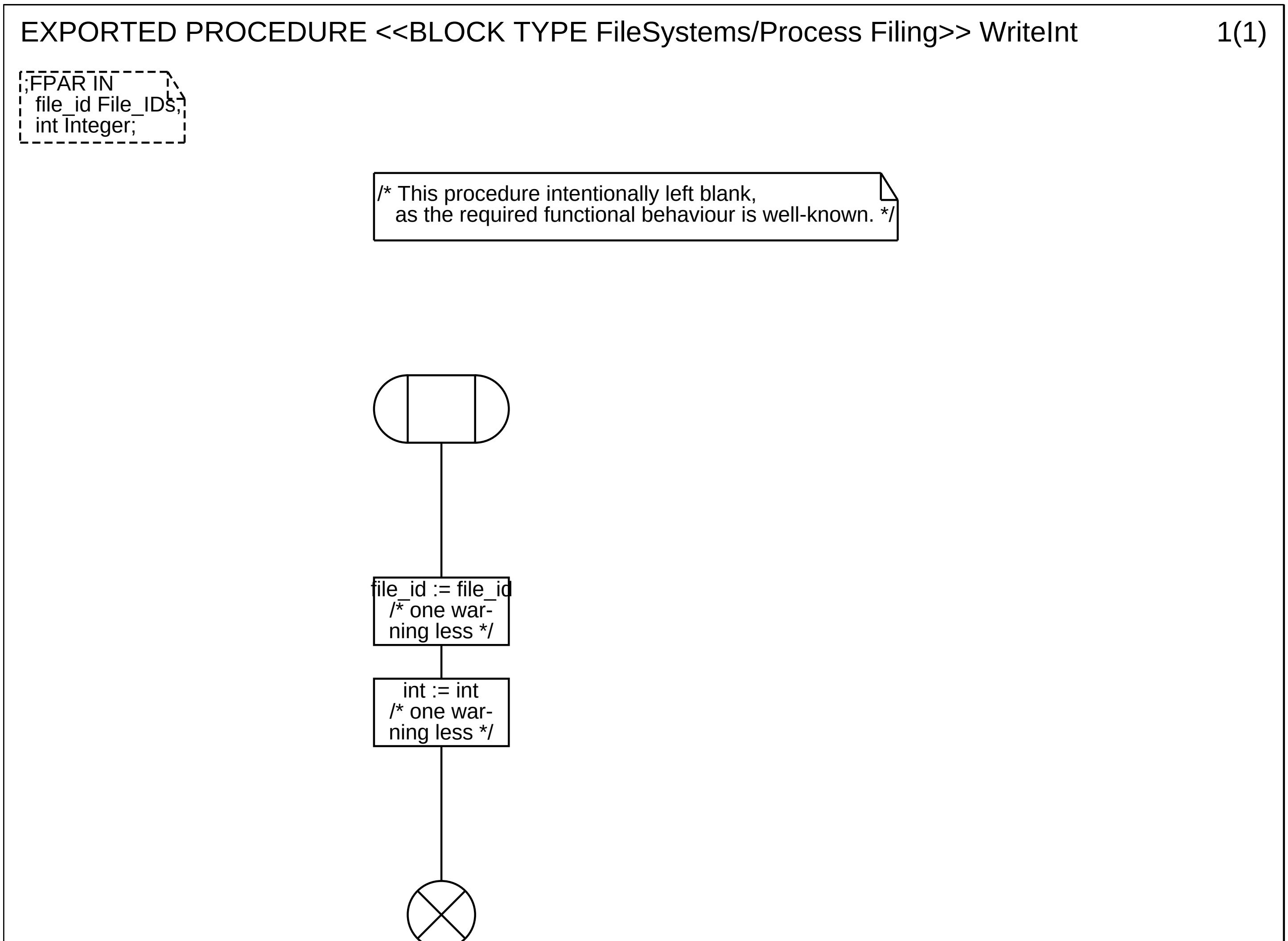
EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> ReadReal




EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> ReadString
1(1)
;FPAR IN
file_id File_IDs;
RETURNS
CharString;
/* This procedure intentionally left blank,
as the required functional behaviour is well-known. */
DCL
string Charstring;
file_id := file_id
/* one war-
ning less */
string

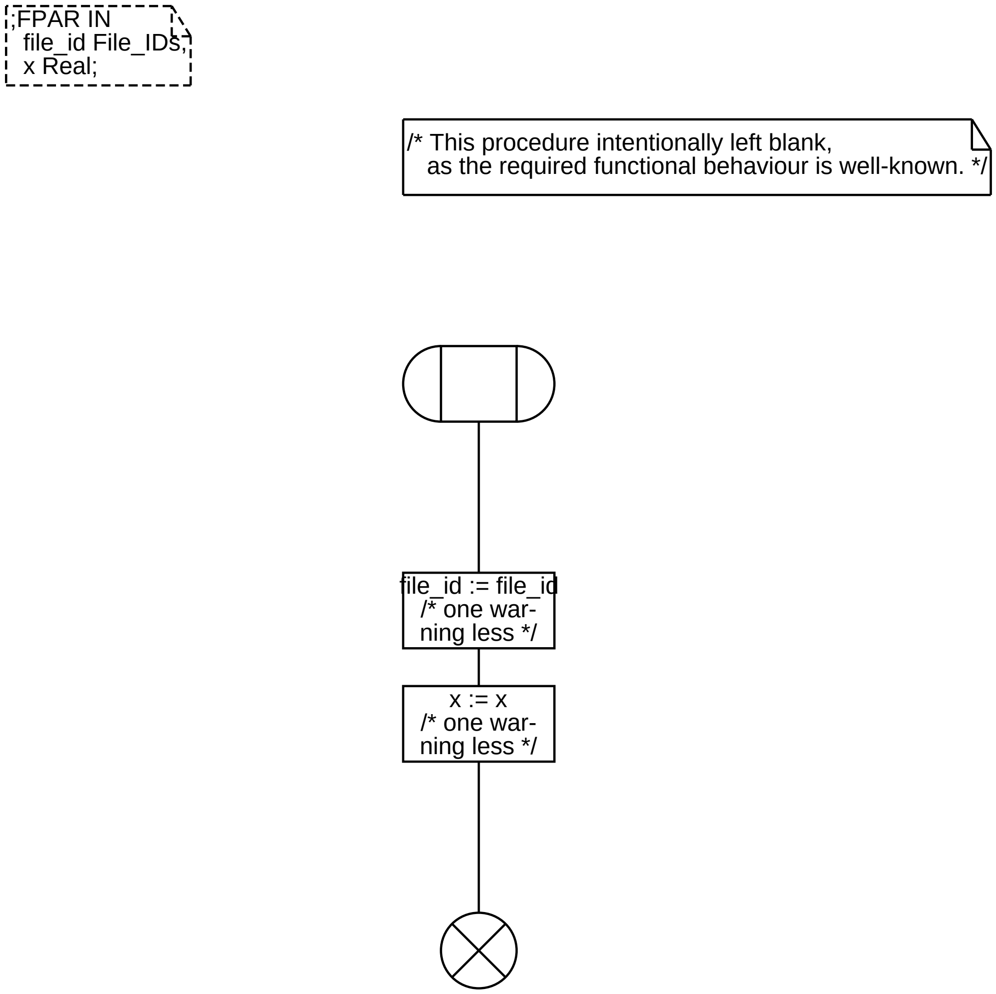
EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> ReadValue 1(1)



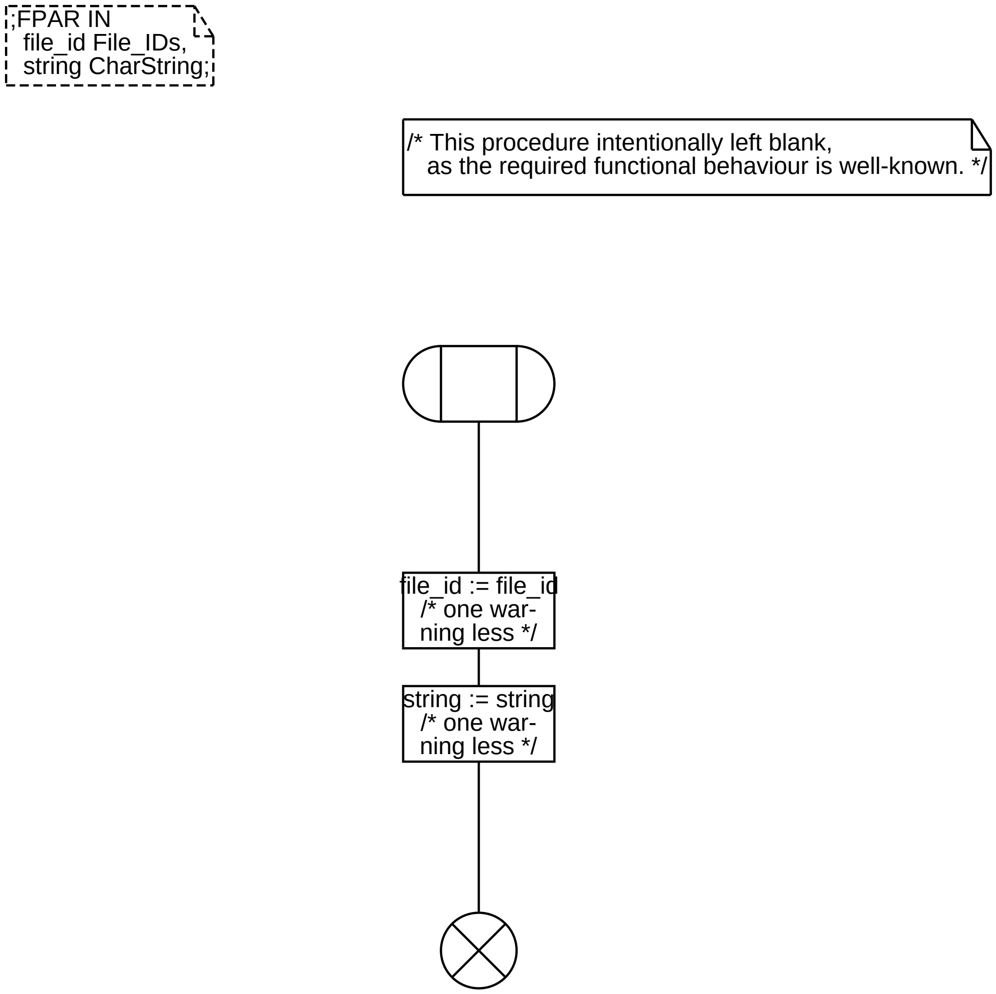


EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> WriteInt
1(1)
;FPAR IN
file_id File_IDs;
int Integer;
/* This procedure intentionally left blank,
as the required functional behaviour is well-known. */
file_id := file_id
/* one warning less */
int := int
/* one warning less */

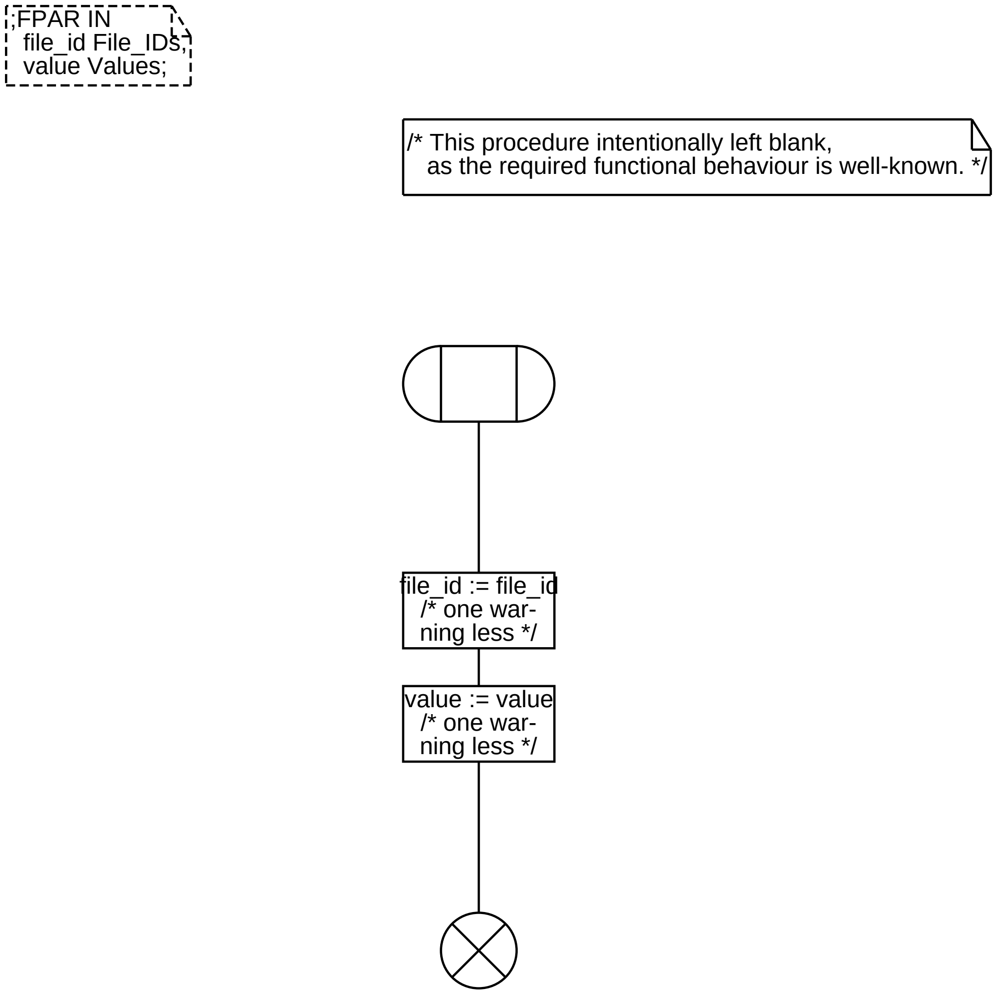
EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> WriteReal 1(1)



EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> WriteString 1(1)



EXPORTED PROCEDURE <<BLOCK TYPE FileSystems/Process Filing>> WriteValue 1(1)



PACKAGE GlobalDataTypes 1(7)

```
NEWTYPE Dependent_variables  /* subtype of Visible_variables representing dependent quantities. */
ENDNEWTYPE Dependent_variables;                  /* When simulating, variables of this type cannot be */
                        /* explicitly set, as their values are the results of their objects' calculations. */
SYNTYPE Dependent_variable_IDs                                  /* the corresponding internal references */
  = IDs
ENDSYNTYPE Dependent_variable_IDs;

SYNTYPE Dependent_variable_names                               /* the corresponding external references */
  = Names
ENDSYNTYPE Dependent_variable_names;

SYNTYPE Durations        /* the set of all the differences between two arbitrary unspecial timestamps */
  = Natural
ENDSYNTYPE Durations;
SYNONYM
  empty_period Durations = 0;

SYNTYPE Event_IDs /* the set of internal event references */
  = Events /* The format of Event_IDs differs from the usual ID format; */
ENDSYNTYPE Event_IDs; /* the type event must be defined in this package as well */

SYNTYPE Event_stream_IDs                                 /* the set of internal event stream references */
  = IDs
ENDSYNTYPE Event_stream_IDs;

SYNTYPE Event_stream_names                                      /* the corresponding external references */
  = Names
ENDSYNTYPE;

NEWTYPE Event_types
  LITERALS
    e_get, e_set, e_sim;
ENDNEWTYPE Event_types;

/* NEWTYPE Events;
   If periodic events are identified here, they have the same period,
   but for simplicity, it is not tested whether the difference of the
   next execution times is a multiple of the period (cf. PROCESS Clock).
   For boolean expressions, AND is expected to have higher priority
   than OR.
*/
```

```
NEWTYPE Events
STRUCT
  event_type Event_types;
  t          Timestamps;
  id         IDs;
  value      Values;
  period     Durations;
  dest       Event_stream_IDs;
ADDING
OPERATORS
  IDEqual: Events, Events -> Boolean;
AXIOMS
  FOR ALL e,f IN Events (
    IDEqual(e,f)==
        event_typeEXTRACT!(e)=event_typeEXTRACT!(f)
      AND
        idEXTRACT!(e)=idEXTRACT!(f)
      AND
        periodEXTRACT!(e)=periodEXTRACT!(f)
      AND
        (periodEXTRACT!(e)=0.0 OR tEXTRACT!(e)=tEXTRACT!(f))
      AND (
          event_typeEXTRACT!(e)=e_get AND
          valueEXTRACT!(e)=valueEXTRACT!(f)
        OR
          event_typeEXTRACT!(e)=e_set AND
          valueEXTRACT!(e)=valueEXTRACT!(f)
        OR
          event_typeEXTRACT!(e)=e_sim);
 )
ENDNEWTYPE Events;

NEWTYPE IDs                    /* supertype comprising all internal reference types */
  STRUCT
    p PId;
    n Natural;
  ADDING
  LITERALS
    id_illegal;
ENDNEWTYPE IDs;
/* Modification: Event_IDs, File_IDs differ */

NEWTYPE Independent_variables      /* subtype of Visible_variables representing independent */
ENDNEWTYPE Independent_variables;
```

PACKAGE GlobalDataTypes

```
SYNTYPE Independent_variable_IDs          /* the corresponding internal references */
  = IDs
ENDSYNTYPE Independent_variable_IDs;

SYNTYPE Independent_variable_names        /* the corresponding external references */
  = Names
ENDSYNTYPE Independent_variable_names;

NEWTYPE Internal_variables        /* generic data type for internal state information of simulation */
ENDNEWTYPE Internal_variables;    /* which is necessary for re-starting; realizes double-buffering */

SYNTYPE Internal_variable_IDs             /* the corresponding internal references */
  = IDs
ENDSYNTYPE Internal_variable_IDs;

SYNTYPE Internal_variable_names           /* the corresponding external references */
  = Names
ENDSYNTYPE Internal_variable_names;

NEWTYPE Messages                  /* the cross product of Visible_variable_names and values */
  STRUCT
    variable Visible_variable_names;
    value Values;
ENDNEWTYPE Messages;

NEWTYPE Names          /* supertype comprising all external reference types; subtype of Strings */
  INHERITS Charstring
  ADDING
  OPERATORS
    ObjectPart: Names -> Names;
    OtherPart: Names -> Names;
    "//": Names, Names -> Names;
    StringToName: Charstring -> Names;
    NameToString: Names -> Charstring;
  AXIOMS
    FOR ALL m, n IN Names (
      OtherPart(n) ==
        IF First(n)='/' THEN Substring(n,2,Length(n))
        ELSE OtherPart(Substring(n,2,Length(n))) FI;
      ObjectPart(n) ==
        Substring(n,1,Length(n)-Length(OtherPart(n))-1);
      m // n == m // TYPE Charstring n;
      NameToString(n) == n;
    );
    FOR ALL c IN Charstrings (
      StringToName(c) == c;
    )
ENDNEWTYPE Names;
```

PACKAGE GlobalDataTypes

```
/* NEWTYPE Names:
   -In Olsen et al.,"Systems Engineering Using SDL-92", the semantics of the Substring operator is
      not defined. It was supposed to be Substring(string,first_position,last_position) with positions
      starting from 1.
   -The empty string must be excluded.
   -For named quantities inside of objects, the slash symbol internally serves to separate object and
      quantity part of the name. Therefore, the slash symbol is allowed in this function only.
*/

SYNTYPE Object_IDs                                  /* the set of internal object references */
  = IDs
ENDSYNTYPE Object_IDs;

SYNTYPE Object_names                                /* the corresponding external references */
  = Names
ENDSYNTYPE Object_names;

NEWTYPE Parameters              /* generic type representing information which is unchangeable */
ENDNEWTYPE Parameters; /*  during a simulation run, e.g. the standard heat output of the radiators */

SYNTYPE Parameter_IDs                               /* the corresponding internal references */
  = IDs
ENDSYNTYPE Parameter_IDs;

SYNTYPE Parameter_names                             /* the corresponding external references */
  = Names
ENDSYNTYPE Parameter_names;

SYNTYPE RPlus                                       /* the set of positive real numbers */
  = Real CONSTANTS > 0
ENDSYNTYPE RPlus;

SYNTYPE Strings                                     /* the set of character strings */
  = CharString
ENDSYNTYPE Strings;

SYNTYPE Timestamps      /* the set of all simulation times. The time unit of simulation time is left ab- */
  = Integer CONSTANTS >= 0   /* stract in the design stage. Depending on the application domain, a */
ENDSYNTYPE Timestamps; /* resolution step could correspond to a millisecond, or to some hours. */

SYNTYPE Timestamps8         /* the set of all simulation times with an additional special timestamp */
  = Integer CONSTANTS >= -1                         /* denoting 'minus infinity' */
ENDSYNTYPE Timestamps8;
SYNONYM
  minus_infinity Timestamps8 = -1;
```

```
SYNTYPE TimestampsStar        /* the set of all simulation times with an additional special timestamp */
  = Integer CONSTANTS >= -1       /* denoting 'as soon as possible'. It does not seem to be possible */
ENDSYNTYPE TimestampsStar;                              /* to write this definition down more sensibly */
SYNONYM                                                          /* (e.g. using inheritance mechanisms) */
  asterisk TimestampsStar = -1;                             /* and still be accepted by the SDT tool */

SYNTYPE Values                    /* supertype comprising all the argument types for the generic types */
  = Integer CONSTANTS >= -1                                     /* Parameters and Visible_variables; */
ENDSYNTYPE Values;                                          /* additionally includes a bottom element */
SYNONYM                                                         /* which is not element of any subtype */
  bottom Values = -1;

NEWTYPE Visible_variables      /* generic data type for that kind of state information of simulation */
ENDNEWTYPE Visible_variables;    /* objects which during the simulation process is visible to the */
  /* event streams; realizes double-buffering; made up of the two subclasses Dependent_- and In- */
                                                                                  /* dependent_variables */
SYNTYPE Visible_variable_IDs                               /* the corresponding internal references */
  = IDs
ENDSYNTYPE Visible_variable_IDs;

SYNTYPE Visible_variable_names                            /* the corresponding external references */
  = Names
ENDSYNTYPE Visible_variable_names;


/* ----------- Construction of list types  ----------- */

NEWTYPE LIST_OF_Event_stream_IDs
  List(Event_stream_IDs)
ENDNEWTYPE LIST_OF_Event_stream_IDs;

NEWTYPE Names_IDs
  STRUCT
    name Names;
    id IDs;
ENDNEWTYPE Names_IDs;

NEWTYPE LIST_OF_Names_IDs
  List(Names_IDs)
ENDNEWTYPE LIST_OF_Names_IDs;

SYNTYPE LIST_OF_Parameter_names_Parameter_IDs
  = LIST_OF_Names_IDs
ENDSYNTYPE LIST_OF_Parameter_names_Parameter_IDs;

SYNTYPE LIST_OF_Visible_variable_names_Visible_variable_IDs
  = LIST_OF_Names_IDs
ENDSYNTYPE LIST_OF_Visible_variable_names_Visible_variable_IDs;
```

```
SYNTYPE LIST_OF_Internal_variable_names_Internal_variable_IDs
  = LIST_OF_Names_IDs
ENDSYNTYPE LIST_OF_Internal_variable_names_Internal_variable_IDs;

GENERATOR List(TYPE Elementtype)
  LITERALS
    nil;
  OPERATORS
    cons:    Elementtype, List -> List;
    car:     List -> Elementtype;
    cdr:     List -> List;
    remove:  Elementtype, List -> List;
    iselem:  Elementtype, List -> BOOLEAN;
  AXIOMS
    FOR ALL l IN List (
      FOR ALL e,f IN Elementtype (
        car(cons(e,l)) == e;
        cdr(cons(e,l)) == l;

        remove(e,nil) == nil;
        remove(e,cons(f,l)) ==
          IF e = f THEN l
          ELSE cons(f,remove(e,l)) FI;

        iselem(e,nil) == FALSE;
        iselem(e,cons(f,l)) == e=f OR iselem(e,l);
        /* Aus unerfindlichen Gruenden akzepitiert der Editor
           bei eingeschalteter Sofortpruefung kein Gleichheits-
           praedikat auf der rechten Seite eines Ausdrucks. Lei-
           der schweigt sich auch die eingebaute Grammatik-Hilfe
           ueber den Aufbau von Axiomen aus. Evtl. muesste die
           zu Elementtype gehoerige Gleichheit verwendet werden.
        */
      )
    )
ENDGENERATOR List;
```

PACKAGE GlobalDataTypes

```
NEWTYPE dummy1          /* only to define "atoi" on Charstring */
OPERATORS
  aton: Charstring -> Natural;
  atoi: Charstring -> Integer;
AXIOMS
  FOR ALL c IN Charstring (
    aton(c // '0') == 10*aton(c);       aton(c // '1') == 10*aton(c)+1;
    aton(c // '2') == 10*aton(c)+2;     aton(c // '3') == 10*aton(c)+3;
    aton(c // '4') == 10*aton(c)+4;     aton(c // '5') == 10*aton(c)+5;
    aton(c // '6') == 10*aton(c)+6;     aton(c // '7') == 10*aton(c)+7;
    aton(c // '8') == 10*aton(c)+8;     aton(c // '9') == 10*aton(c)+9;

    atoi('-' // c) == -aton(c);         atoi('1' // c) == aton('1' // c);
    atoi('2' // c) == aton('2' // c);   atoi('3' // c) == aton('3' // c);
    atoi('4' // c) == aton('4' // c);   atoi('5' // c) == aton('5' // c);
    atoi('6' // c) == aton('6' // c);   atoi('7' // c) == aton('7' // c);
    atoi('8' // c) == aton('8' // c);   atoi('9' // c) == aton('9' // c);
  )
ENDNEWTYPE dummy1;

NEWTYPE dummy2          /* only to define "atof" on Charstring */
OPERATORS       /* and conversion operations Timestamps -> */
  atof: Charstring -> Real;                    /* RPlus -> Durations */
  float: Timestamps -> RPlus;
  dur: RPlus -> Duration;                 /*
AXIOMS
   left to the reader as night exercise ... */
ENDNEWTYPE dummy2;
```

```
USE GlobalDataTypes;
USE ExternalSignals;
USE FilingSystem;
```

PACKAGE InternalSignals


```
/* Channel A */
SIGNAL
  register_object_id(Object_names, PId,
    LIST_OF_Parameter_names_Parameter_IDs,
    LIST_OF_Visible_variable_names_Visible_variable_IDs,
    LIST_OF_Internal_variable_names_Internal_variable_IDs, Durations),
  schedule_get_event(Timestamps, Visible_variable_IDs, Durations, Event_stream_IDs),
  schedule_set_event(Timestamps, Visible_variable_IDs, Durations, Values),
  output_internal(Timestamps, Visible_Variable_IDs, Values, LIST_OF_Event_stream_IDs),
  /*output_error(Timestamps, Visible_Variable_IDs, Values, Durations, Strings),*/
  get_name(IDs), name(IDs, Names);

SIGNALLIST
  Schedule = schedule_get_event, schedule_set_event;

SIGNALLIST
  Schedule_return = scheduled_event, schedule_error;

SIGNALLIST
  ID_time_up = register_object_id, get_id, get_name, simulation_time;

SIGNALLIST
  ID_time_down = id, name, get_simulation_time;


/* Channel B */
SIGNAL
  retrieve_dump_from(File_IDs), dump_values_to(File_IDs),
  dump_anything_to(File_IDs), dump_done;

SIGNALLIST
  Initialize_kernel =
    initialize, set_parameter, set_time_mode_fast, set_time_mode,
    set_simulation_start, retrieve_dump_from, get_ready;

SIGNALLIST
  Control_kernel =
    set_independent_variable, set_simulation_end, unset_simulation_end,
    set_halting_point, unset_halting_point, stop_simulation,
    resume_simulation, dump_values_to, dump_anything_to;

SIGNALLIST
  Kernel_return =
    dump_done, (Simulator_return);


/* Channel C */
SIGNAL
  modify, dump_OK, dump_different;

SIGNALLIST
  Control_interface = initialize, modify, retrieve_dump_from, dump_anything_to, finished;

SIGNALLIST
  Interface_return = simulation_warning, simulation_error, dump_OK, dump_different;
```

```
USE GlobalDataTypes;
USE ExternalSignals;
USE FilingSystem;
```

PACKAGE InternalSignals


```
/* Kernel */

SIGNAL
  get_actual_events(Timestamps), actual_events(Event_sets),
  advance_simulation_time, no_objects(Natural);

SIGNALLIST
  Dump =
    retrieve_dump_from, dump_anything_to, dump_values_to;

SIGNALLIST
  PassThruSO =
    empty_output_configuration, id, set_parameter, set_visible_variable,
    set_independent_variable, set_max_period, add_output_configuration,
    get_anything, get_values;


SIGNALLIST                                /* Splitting up Control_kernel */
  Control_handling =
    set_independent_variable;

SIGNALLIST
  Control_clock =
    set_simulation_end, unset_simulation_end, set_halting_point,
    unset_halting_point, stop_simulation, resume_simulation,
    dump_values_to, dump_anything_to;

SIGNALLIST                                /* Splitting up Kernel_return */
  Handling_return =
    simulation_error;

SIGNALLIST
  Clock_return =
    simulation_stopped, simulation_slacked, simulation_OK,
    simulation_warning, simulation_error, dump_done;

SIGNALLIST                                /* Splitting up Initialize_kernel */
  Initialize_handling =
    initialize, set_parameter, get_ready;

SIGNALLIST
  Initialize_clock =
    set_time_mode_fast, set_time_mode,
    set_simulation_start, retrieve_dump_from;

SIGNALLIST                                /* Splitting up Initialize_return */
  Initialize_h_return =
    initialize_OK, initialize_error,
    get_ready_OK, get_ready_error;
```

```
/* Kernel_return: The remaining
SIGNAL
  simulation_warning;
is issued by PassThru. */

/* Initialize_return: The remaining
SIGNAL
  modify_warning;
is issued by PassThru. */
```

```
USE GlobalDataTypes;
USE ExternalSignals;
USE FilingSystem;
```

PACKAGE InternalSignals

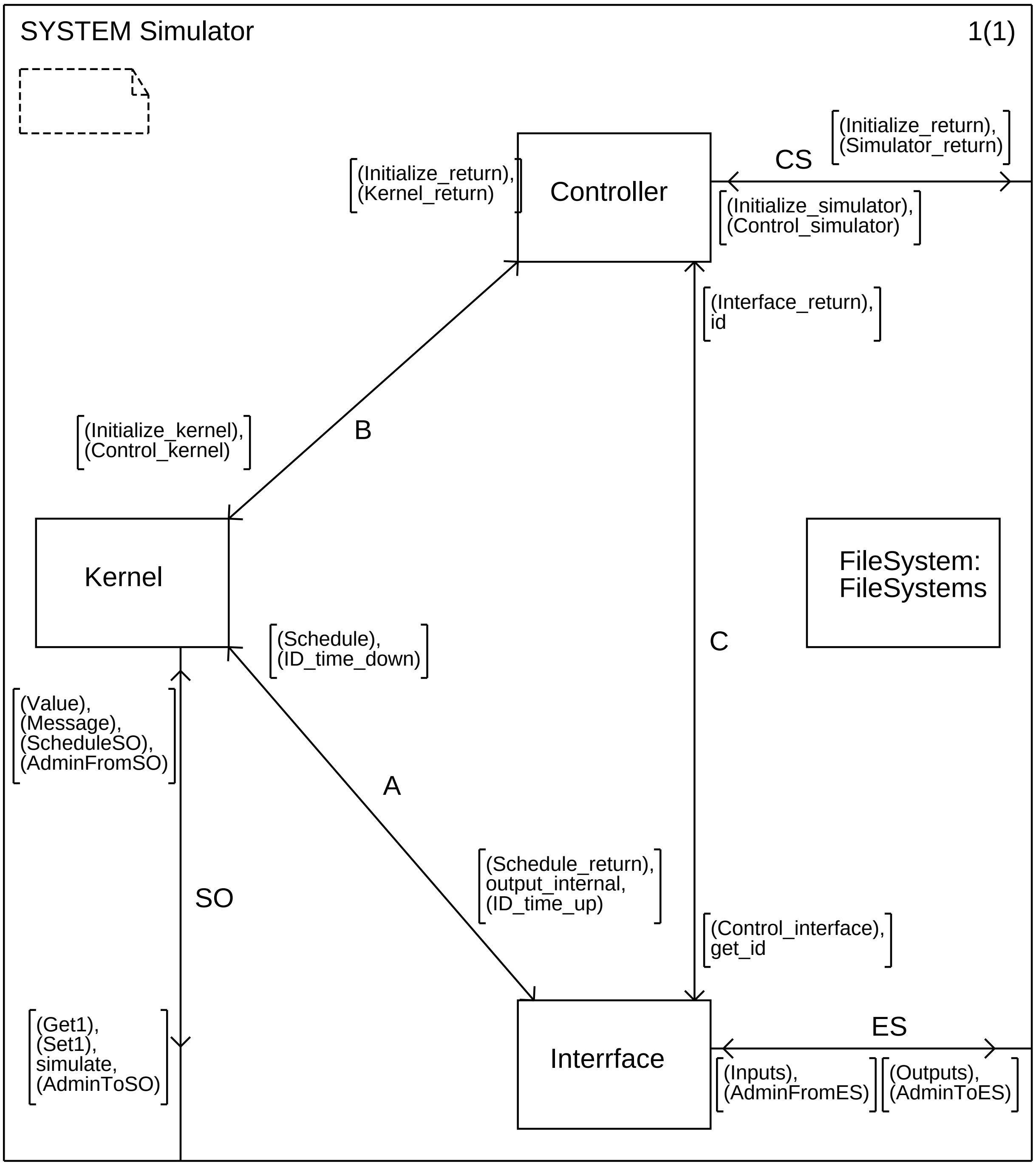

```
 NEWTYPE Event_sets
  List(Events)
ADDING
OPERATORS
  InsertEvent:     Events, Event_sets -> Event_sets;
  RemoveEvent:     Events, Event_sets -> Event_sets;
  RemoveObject:    PId, Event_sets -> Event_sets;
  AlterEvent:      Events, Durations, Event_sets -> Event_sets;
  StampEvents:     Timestamps, Event_sets -> Event_sets;
AXIOMS
  FOR ALL e,f IN Events (
  FOR ALL s IN Event_sets (
  FOR ALL t IN Timestamps (
  FOR ALL d IN Durations (
  FOR ALL o IN PId (
    InsertEvent(e,nil) == cons(e,nil);  /* Insert event into event collection which */
    InsertEvent(e,cons(f,s)) ==   /* is ordered according to components event!t */
      IF tEXTRACT(e)<=tEXTRACT(f) THEN cons(e,cons(f,s))
      ELSE cons(f,InsertEvent(e,s)) FI;

    RemoveEvent(e,nil) == nil;      /* Remove an event from the event collection */
    RemoveEvent(e,cons(f,s)) ==
      IF IDEqual(e,f) THEN s
      ELSE cons(f,RemoveEvent(e,s)) FI;

    RemoveObject(o,nil) == nil;             /* Remove an object's event from a set of */
    RemoveObject(o,cons(f,s)) ==          /* events. Used in EventHandling when */
      IF o=pEXTRACT!(idEXTRACT!(f)) THEN s           /* activated objects return */
      ELSE cons(f,RemoveObject(o,s)) FI;                     /* signal object_finished */

    AlterEvent(e,d,nil) == nil;                              /* Change the period of an event */
    AlterEvent(e,d,cons(f,s)) ==                                          /* in the event collection. */
      IF IDEqual(e,f) THEN
        cons(EventsMAKE!(event_typeEXTRACT!(f),tEXTRACT!(f),
          idEXTRACT!(f),valueEXTRACT!(f),d,destEXTRACT!(f)),s)
      ELSE cons(f,AlterEvent(e,s)) FI;

    StampEvents(t,nil) == nil;                 /* When getting ready, discard outdated */
    StampEvents(t,cons(e,s)) ==           /* events and give reasonable starting- */
      IF periodEXTRACT!(e)=0.0 THEN                          /* time to periodic ones. */
        IF tEXTRACT!(e)<t THEN StampEvents(t,s)
        ELSE cons(e,StampEvents(t,s)) FI
      ELSE
        cons(EventsMAKE!(event_typeEXTRACT!(e),
          tEXTRACT!(e)+periodEXTRACT!(e)
            *(1+(t-tEXTRACT!(e))/periodEXTRACT!(e)),
          idEXTRACT!(e),valueEXTRACT!(e),periodEXTRACT!(e),
          destEXTRACT!(e)),StampEvents(t,s))
      FI;
    /* Remember the starting-point t in simulation time is changeable.
       t_new:=t_old+k*period with t_new>t and t_new-period<=t
       => (t-t_old)/period < k <= (t-t_old)/period+1
       => k=floor((t-t_old)/p)+1 */
 )))))
ENDNEWTYPE Event_Sets;
```

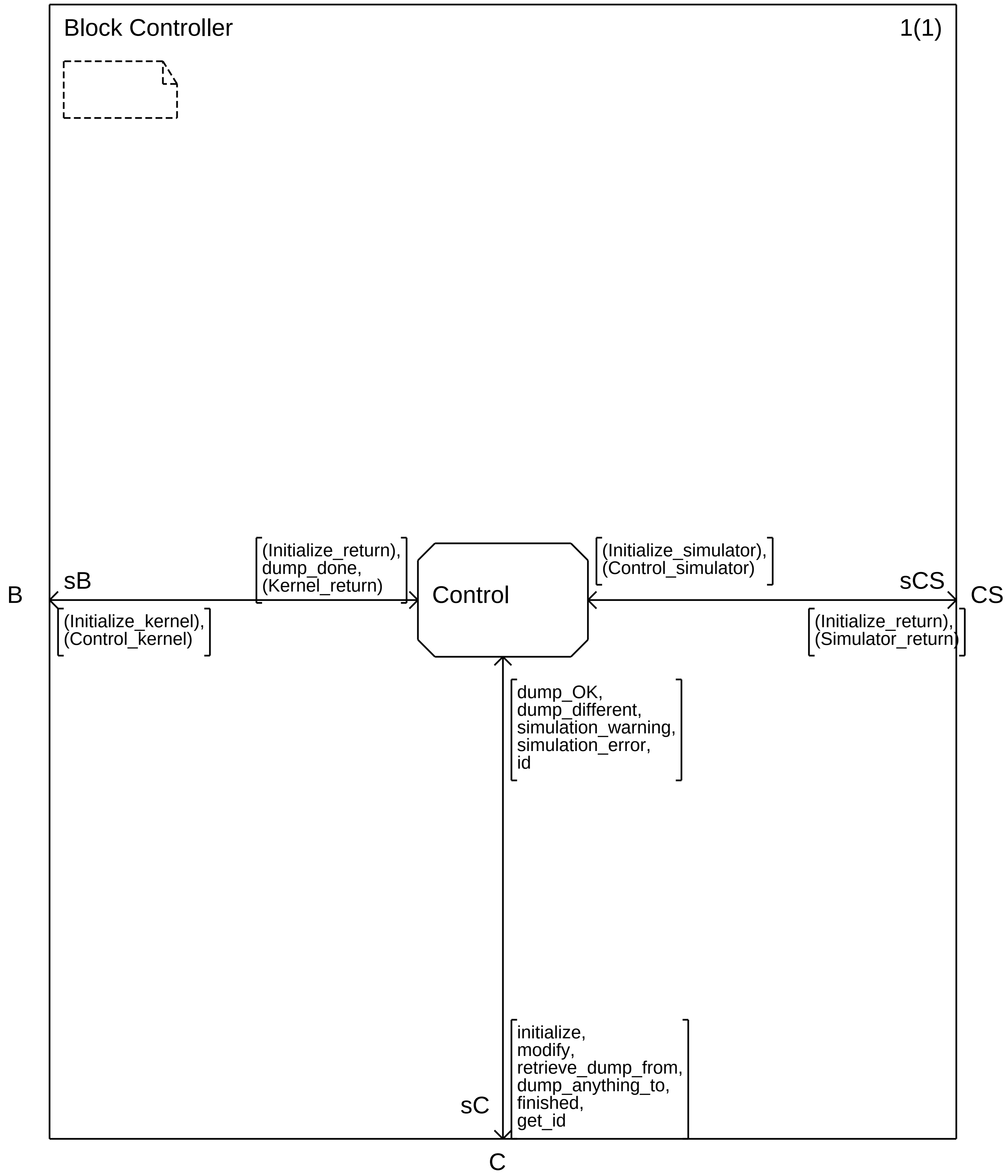
```
USE GlobalDataTypes;
USE ExternalSignals;
USE FilingSystem;
```

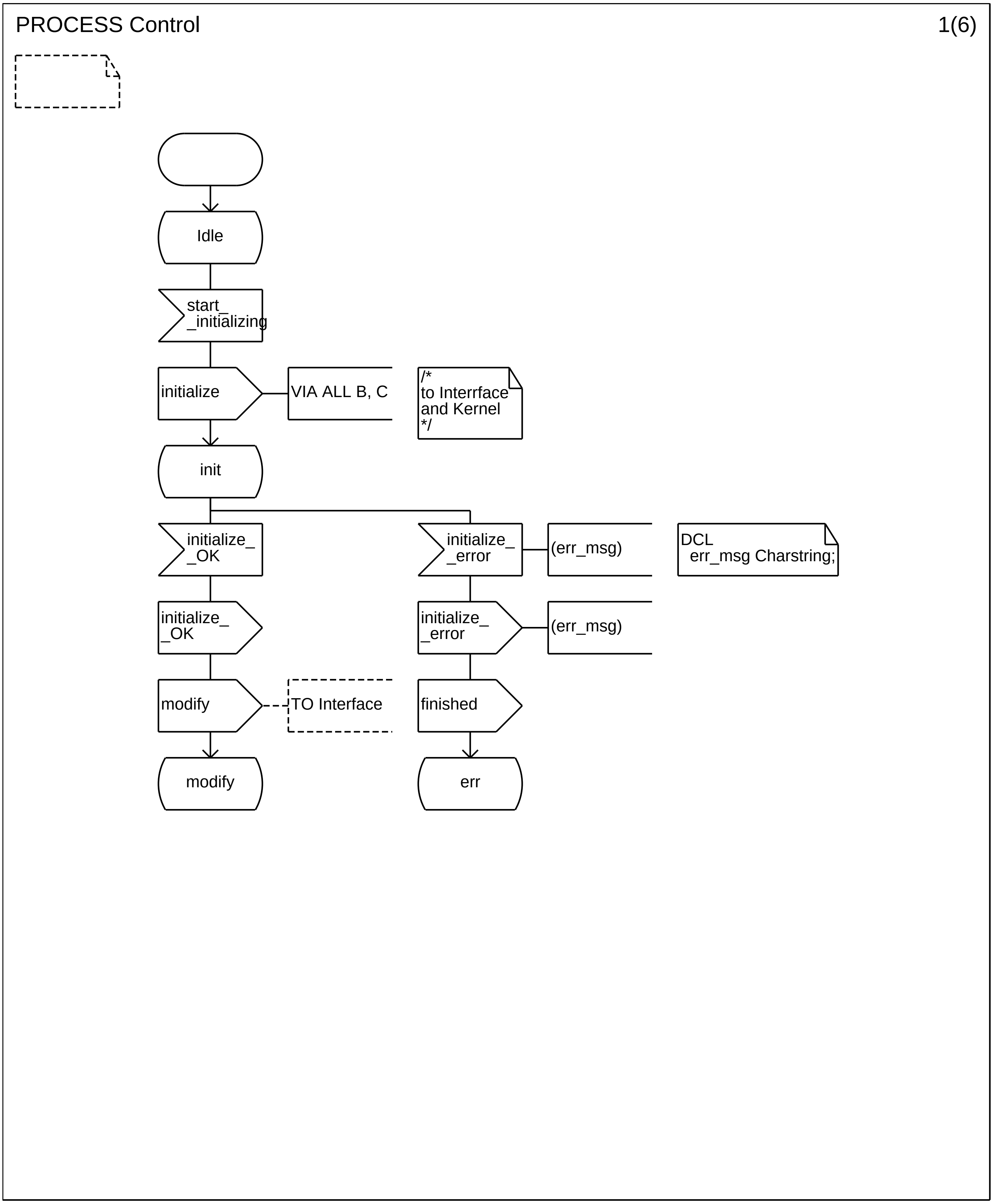


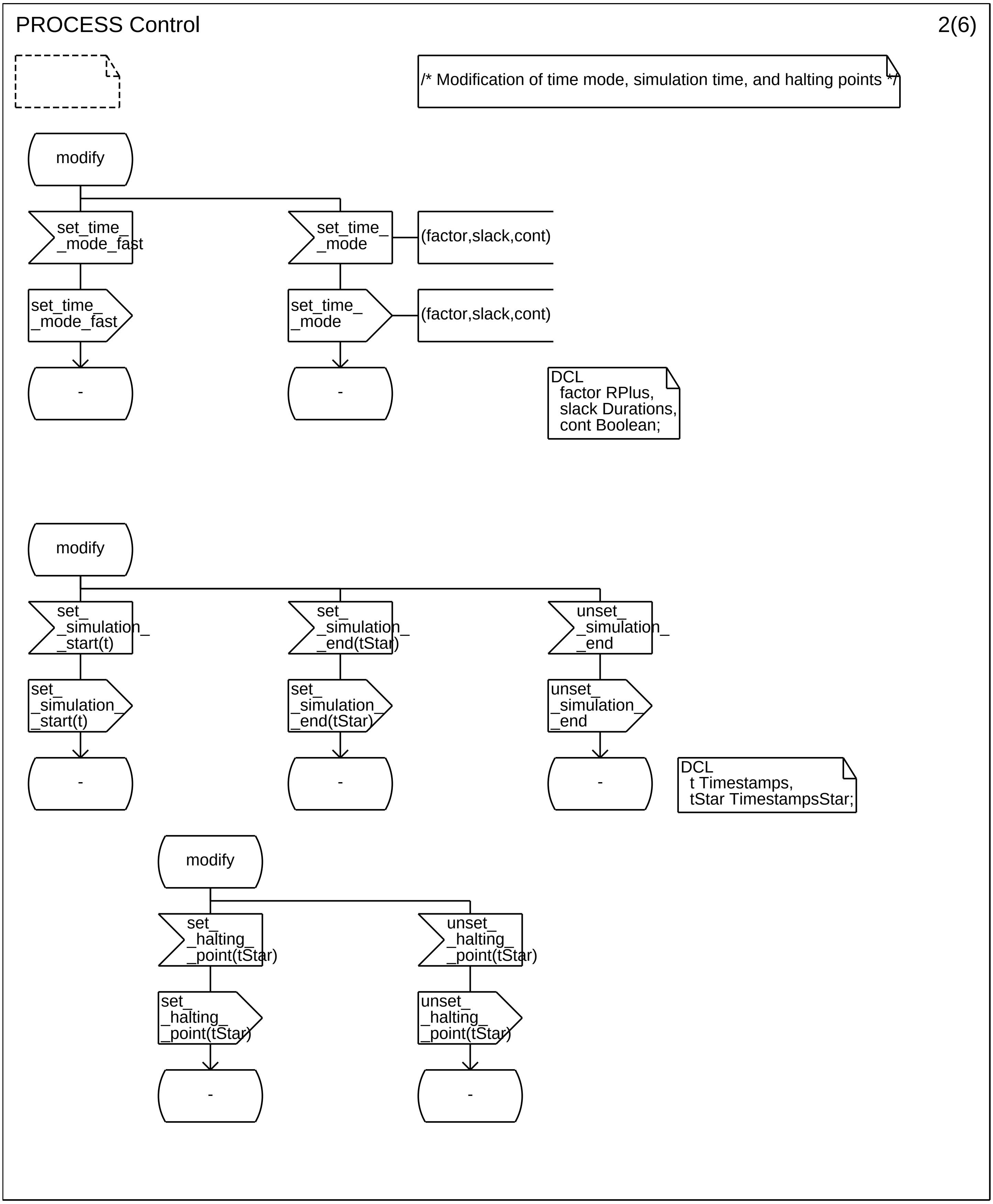
```
NEWTYPE Event_sets_cont /* Dummy type due to page break */
OPERATORS
  ActualEvents:     Timestamps, Event_sets -> Event_sets;
  NonactualEvents:  Timestamps, Event_sets -> Event_sets;
  ReinsertEvents:   Event_sets, Event_sets -> Event_sets;
AXIOMS
  FOR ALL e IN Events (
  FOR ALL r,s IN Event_sets (
  FOR ALL t IN Timestamps (
    ActualEvents(t,nil) == nil;                   /* Get all the events with timestamp <=t */
    ActualEvents(t,cons(e,s)) ==
      IF t<tEXTRACT!(e) THEN nil                  /* Remember that the list is sorted. */
      ELSE cons(e,ActualEvents(t,s)) FI;

    NonctualEvents(t,nil) == nil;        /* Discard all the events with timestamp <=t */
    NonctualEvents(t,cons(e,s)) ==
      IF t<tEXTRACT!(e) THEN cons(e,s)
      ELSE NonactualEvents(t,s) FI;

    Reinsert(nil,s) == nil;                                  /* Reinsert periodic events */
    Reinsert(cons(e,r),s) ==                                /* into the event collection */
      IF periodEXTRACT!(e)=0.0 THEN Reinsert(r,s)
      ELSE
        InsertEvent(EventsMAKE!(event_typeEXTRACT!(e),
          tEXTRACT!(e)+periodEXTRACT!(e),idEXTRACT!(e),
          valueEXTRACT!(e),periodEXTRACT!(e),destEXTRACT!(e)),
          Reinsert(r,s))
      FI;
  )))
ENDNEWTYPE Event_Sets_cont;

GENERATOR SortedList(TYPE ElementType)
  List(ElementType)
  ADDING
  OPERATORS
    Insert: ElementType, SortedList -> SortedList;
  AXIOMS
    FOR ALL e,f in ElementType (
      FOR ALL l IN SortedList (
        Insert(e,nil) == cons(e,nil);                        /* Insert one element */
        Insert(e,cons(f,l)) ==
          IF e<=f THEN cons(e,cons(f,l))
          ELSE cons(f,Insert(e,l)) FI;
      )
    )
ENDGENERATOR SortedList;

NEWTYPE LIST_OF_Timestamps
  SortedList(Timestamps)
ENDNEWTYPE LIST_OF_Timestamps;
```

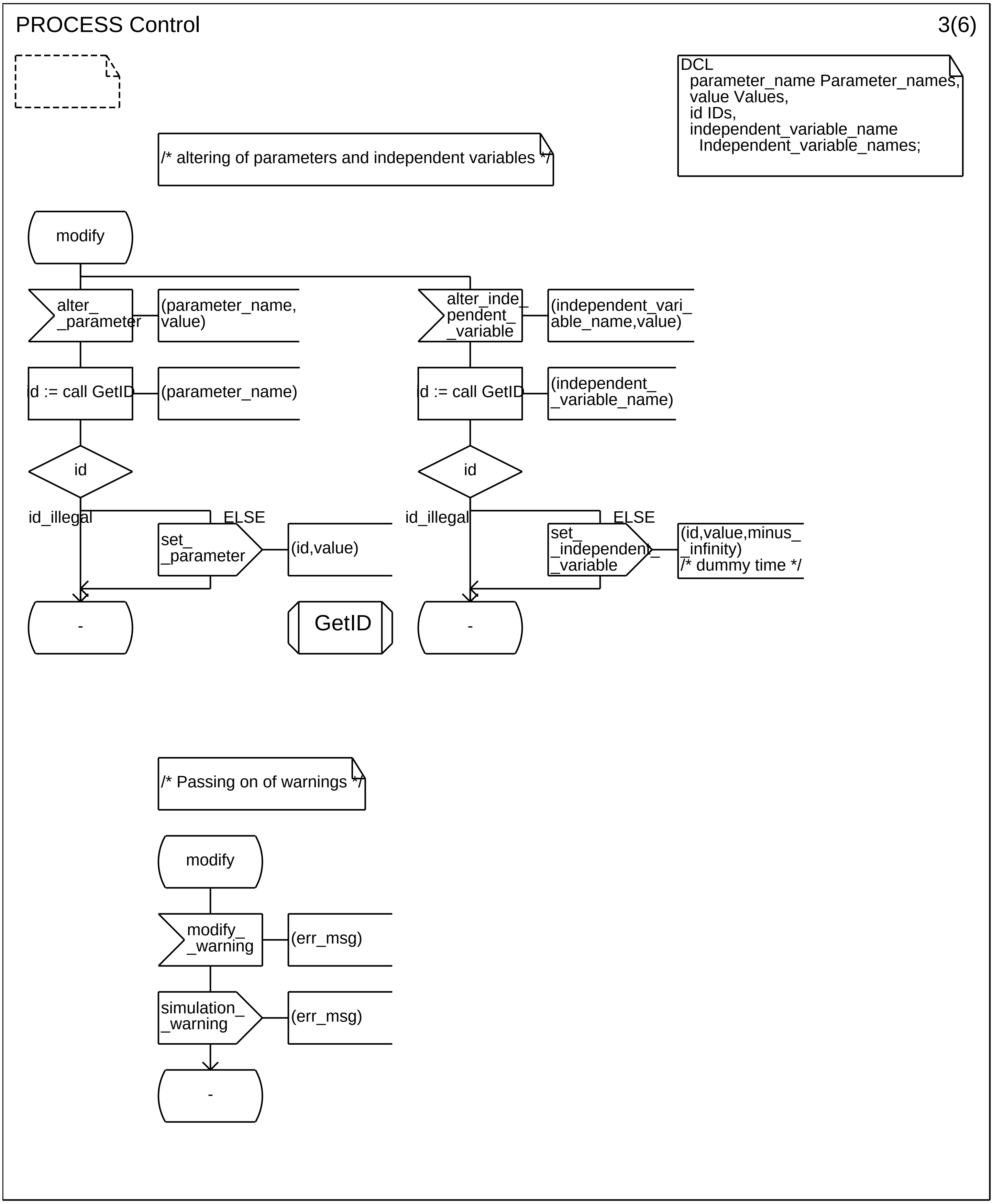
USE GlobalDataTypes;
USE ExternalSignals;
USE InternalSignals;
USE FilingSystem;

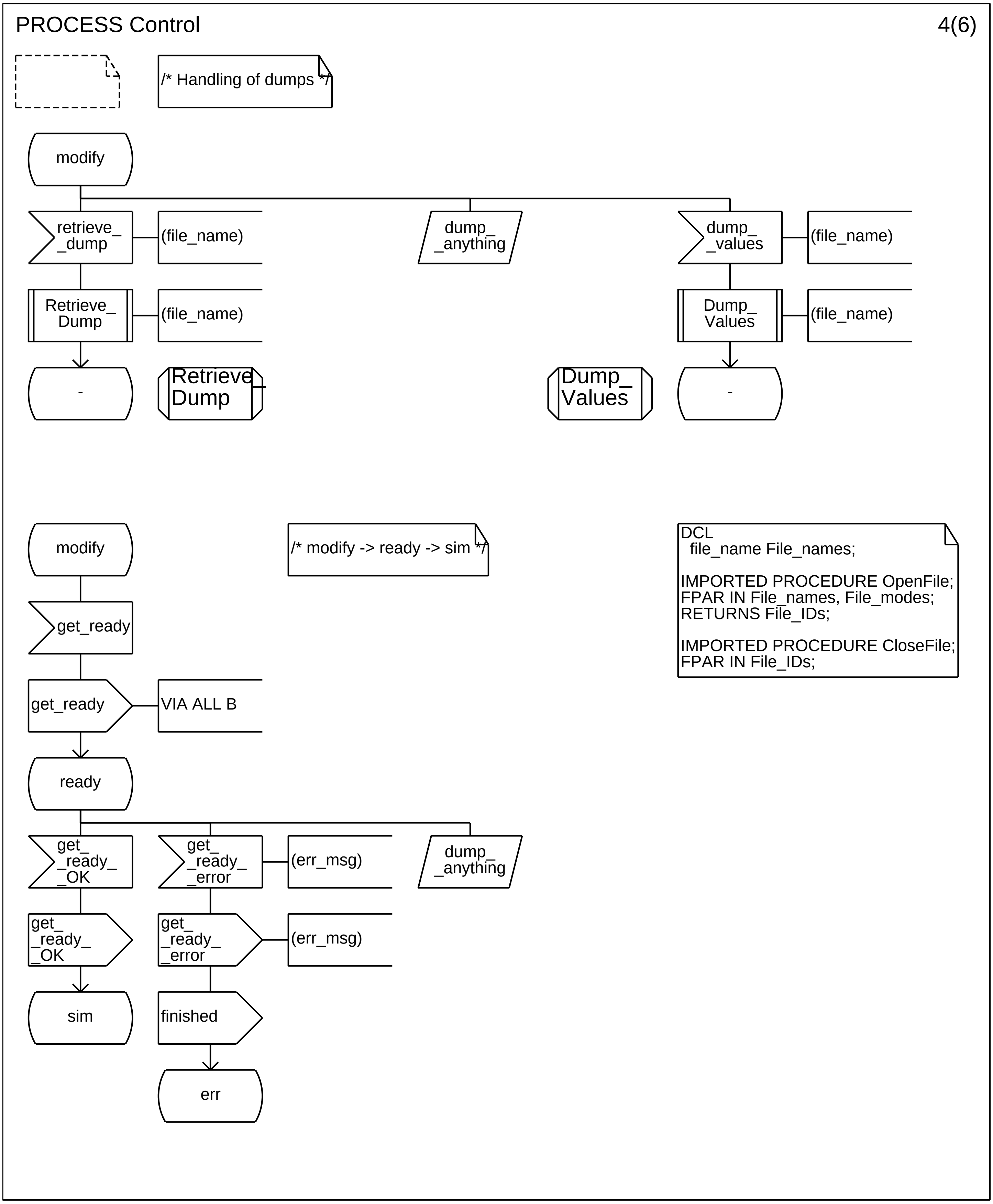


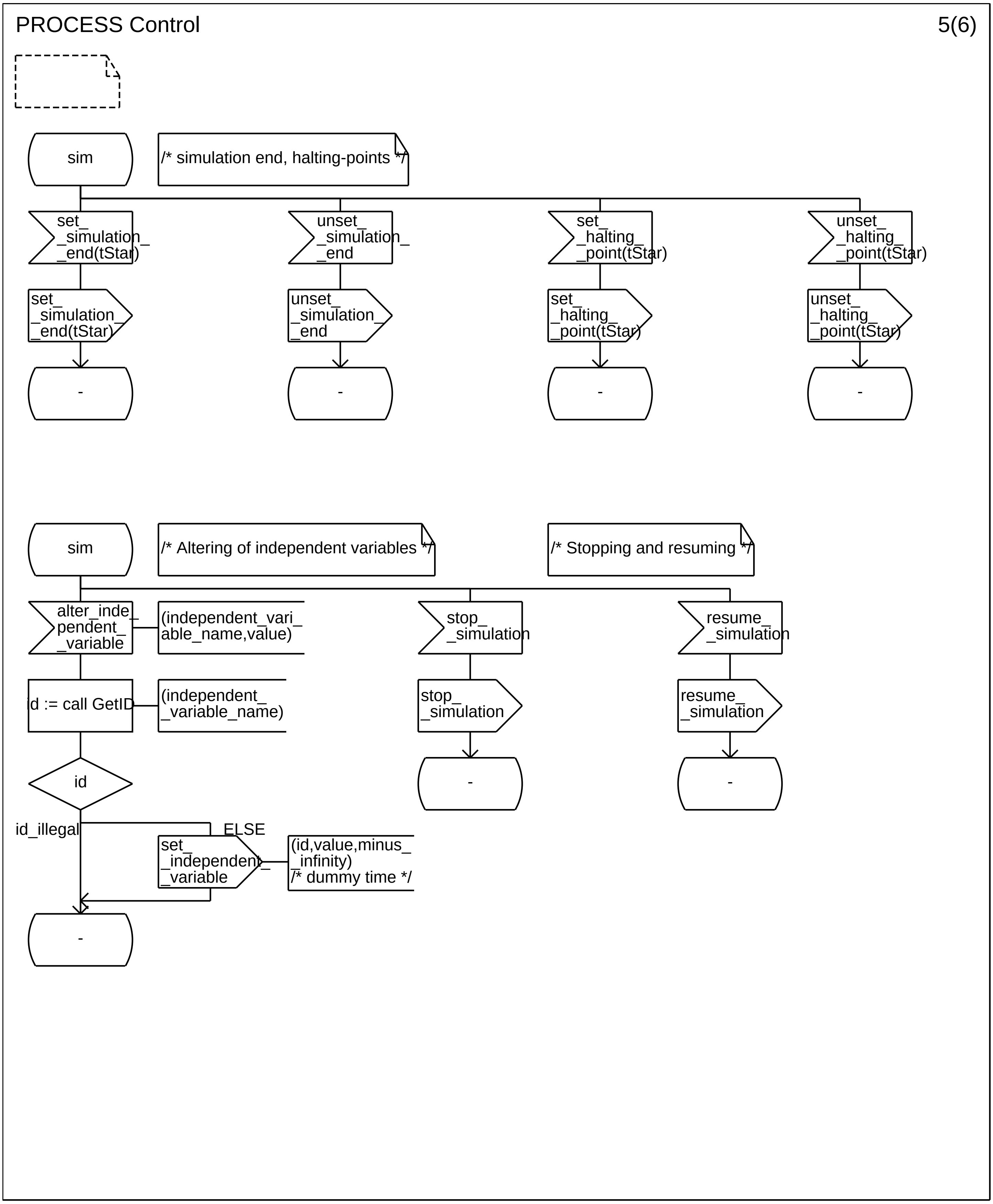

Block Controller
1(1)
sB
B
(Initialize_return),
dump_done,
(Kernel_return)
(Initialize_kernel),
(Control_kernel)
Control
(Initialize_simulator),
(Control_simulator)
sCS
CS
(Initialize_return),
(Simulator_return)
dump_OK,
dump_different,
simulation_warning,
simulation_error,
id
initialize,
modify,
retrieve_dump_from,
dump_anything_to,
finished,
get_id
sC
C

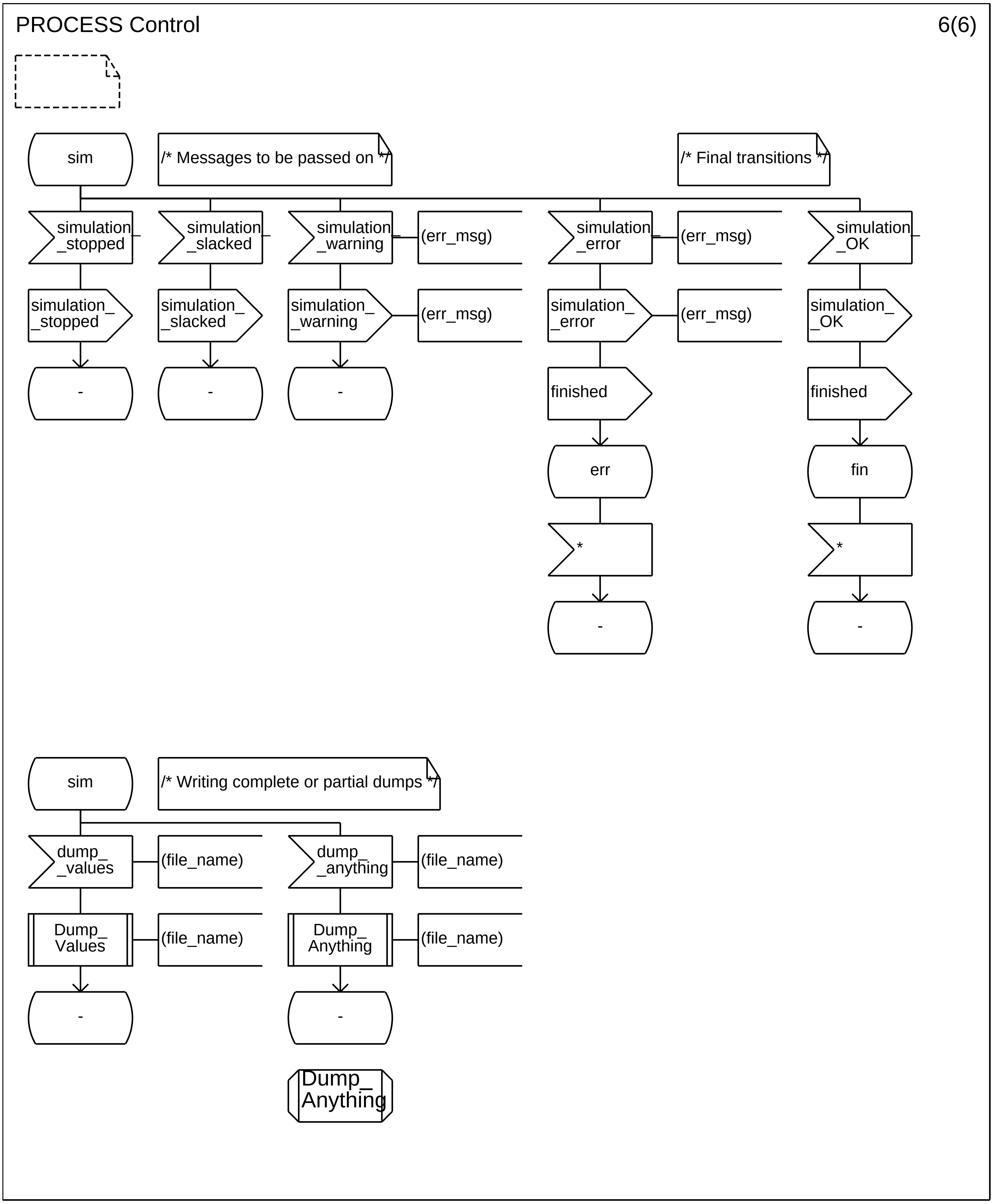

PROCESS Control
1(6)
Idle
start_
_initializing
initialize
VIA ALL B, C
/*
to Interrface
and Kernel
*/
init
initialize_
_OK
initialize_
_error
(err_msg)
DCL
err_msg Charstring;
initialize_
_OK
initialize_
_error
(err_msg)
modify
TO Interface
finished
modify
err


PROCESS Control
2(6)
/* Modification of time mode, simulation time, and halting points */
modify
set_time_
_mode_fast
set_time_
_mode
(factor,slack,cont)
set_time_
_mode_fast
set_time_
_mode
(factor,slack,cont)
-
-
DCL
factor RPlus,
slack Durations,
cont Boolean;
modify
set_
simulation
_start(t)
set_
simulation
_end(tStar)
unset_
simulation
_end
set_
simulation
_start(t)
set_
simulation
_end(tStar)
unset_
simulation
_end
-
-
-
DCL
t Timestamps,
tStar TimestampsStar;
modify
set_
halting
_point(tStar)
unset_
halting
_point(tStar)
set_
halting
_point(tStar)
unset_
halting
_point(tStar)
-
-

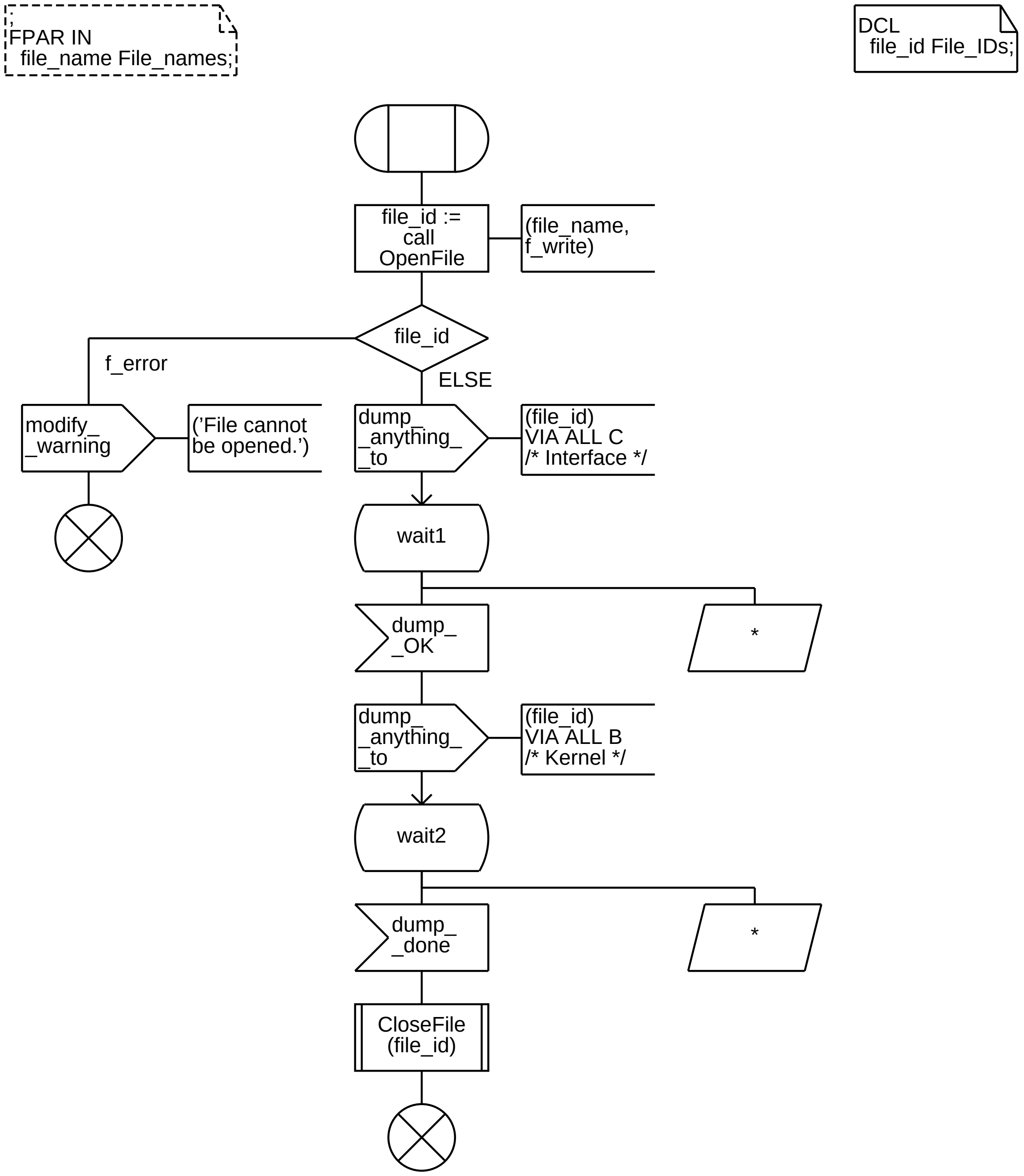

PROCESS Control
3(6)
DCL
parameter_name Parameter_names,
value Values,
id IDs,
independent_variable_name
Independent_variable_names;
/* altering of parameters and independent variables */
modify
alter_
_parameter
(parameter_name,
value)
id := call GetID
(parameter_name)
id
id_illegal
ELSE
set_
_parameter
(id,value)
-
GetID
alter_inde_
pendent_
_variable
(independent_vari_
able_name,value)
id := call GetID
(independent_
_variable_name)
id
id_illegal
ELSE
set_
independent
_variable
(id,value,minus_
_infinity)
/* dummy time */
-
/* Passing on of warnings */
modify
modify_
_warning
(err_msg)
simulation_
_warning
(err_msg)
-


PROCESS Control
4(6)
/* Handling of dumps */
modify
retrieve_
_dump
(file_name)
dump_
_anything
dump_
_values
(file_name)
Retrieve_
Dump
(file_name)
Dump_
Values
(file_name)
-
Retrieve_
Dump
Dump_
Values
-
modify
/* modify -> ready -> sim */
DCL
file_name File_names;
IMPORTED PROCEDURE OpenFile;
FPAR IN File_names, File_modes;
RETURNS File_IDs;
IMPORTED PROCEDURE CloseFile;
FPAR IN File_IDs;
get_ready
get_ready
VIA ALL B
ready
get_
ready
_OK
get_
ready
_error
(err_msg)
dump_
_anything
get_
ready
_OK
get_
ready
_error
(err_msg)
sim
finished
err

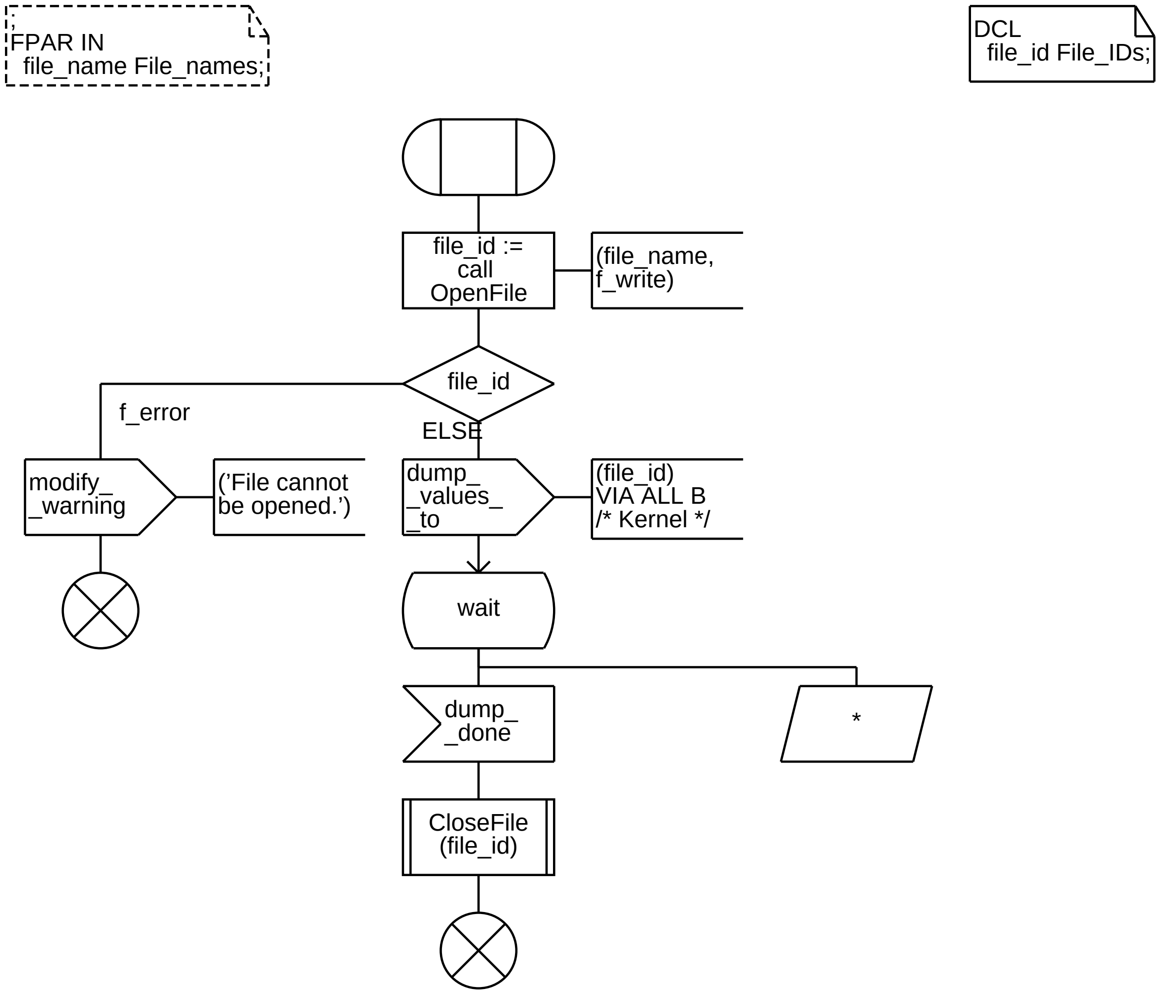

PROCESS Control
5(6)
sim
/* simulation end, halting-points */
set_ _simulation_ _end(tStar)
set_ _simulation_ _end(tStar)
-
unset_ _simulation_ _end
unset_ _simulation_ _end
-
set_ _halting_ _point(tStar)
set_ _halting_ _point(tStar)
-
unset_ _halting_ _point(tStar)
unset_ _halting_ _point(tStar)
-
sim
/* Altering of independent variables */
/* Stopping and resuming */
alter_inde_ pendent_ _variable
(independent_vari_ able_name,value)
id := call GetID
(independent_ _variable_name)
id
id_illegal
ELSE
set_ _independent_ _variable
(id,value,minus_ _infinity) /* dummy time */
-
stop_ _simulation
stop_ _simulation
-
resume_ _simulation
resume_ _simulation
-

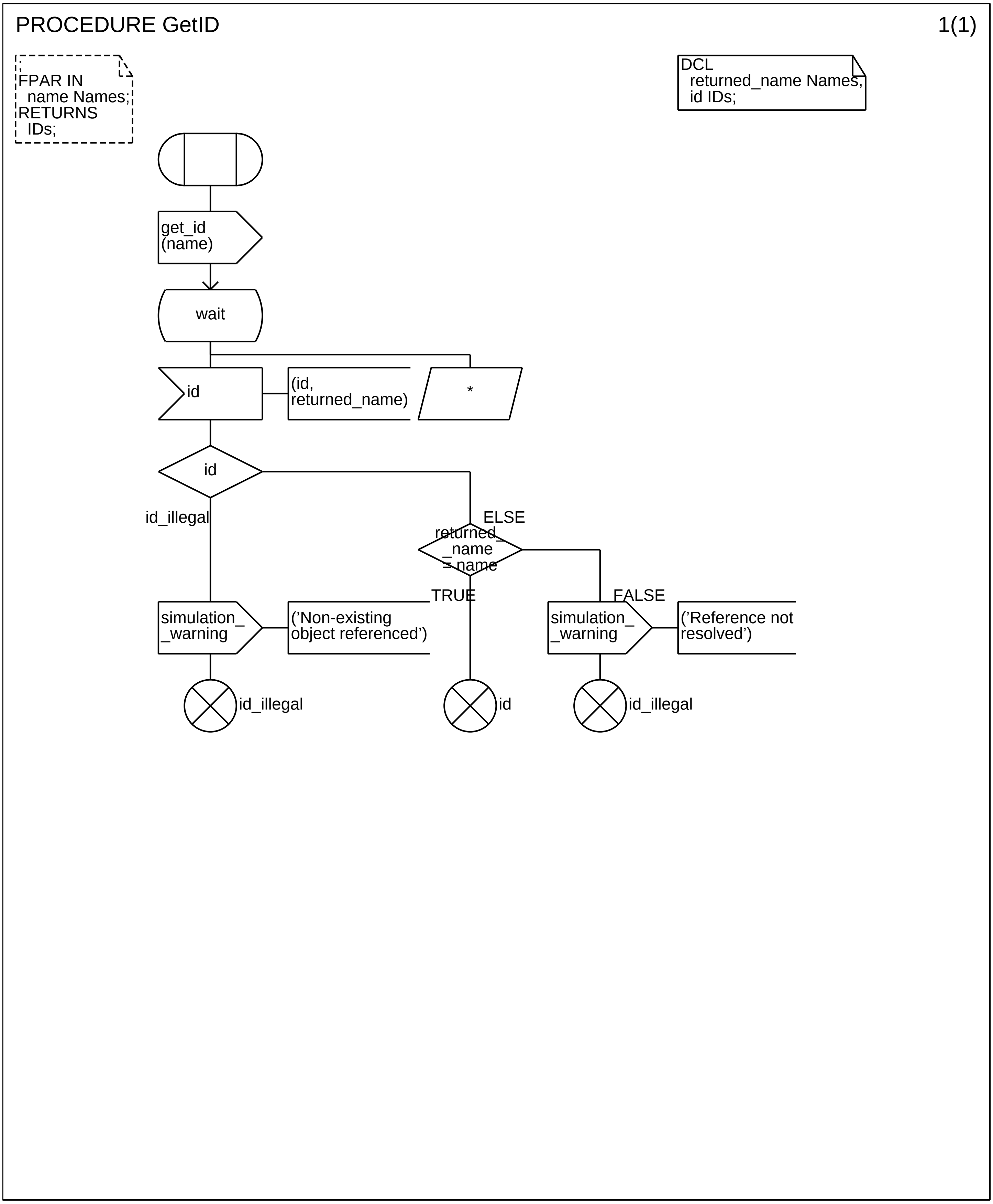

PROCESS Control
6(6)
sim
/* Messages to be passed on */
/* Final transitions */
simulation_ _stopped
simulation_ _slacked
simulation_ _warning
(err_msg)
simulation_ _error
(err_msg)
simulation_ _OK
simulation_ _stopped
simulation_ _slacked
simulation_ _warning
(err_msg)
simulation_ _error
(err_msg)
simulation_ _OK
-
-
-
finished
finished
err
fin
*
*
-
-
sim
/* Writing complete or partial dumps */
dump_ _values
(file_name)
dump_ _anything
(file_name)
Dump_ Values
(file_name)
Dump_ Anything
(file_name)
-
-
Dump_ Anything

PROCEDURE <<PROCESS Control>> DumpAnything 1(1)



PROCEDURE <<PROCESS Control>> DumpValues 1(1)



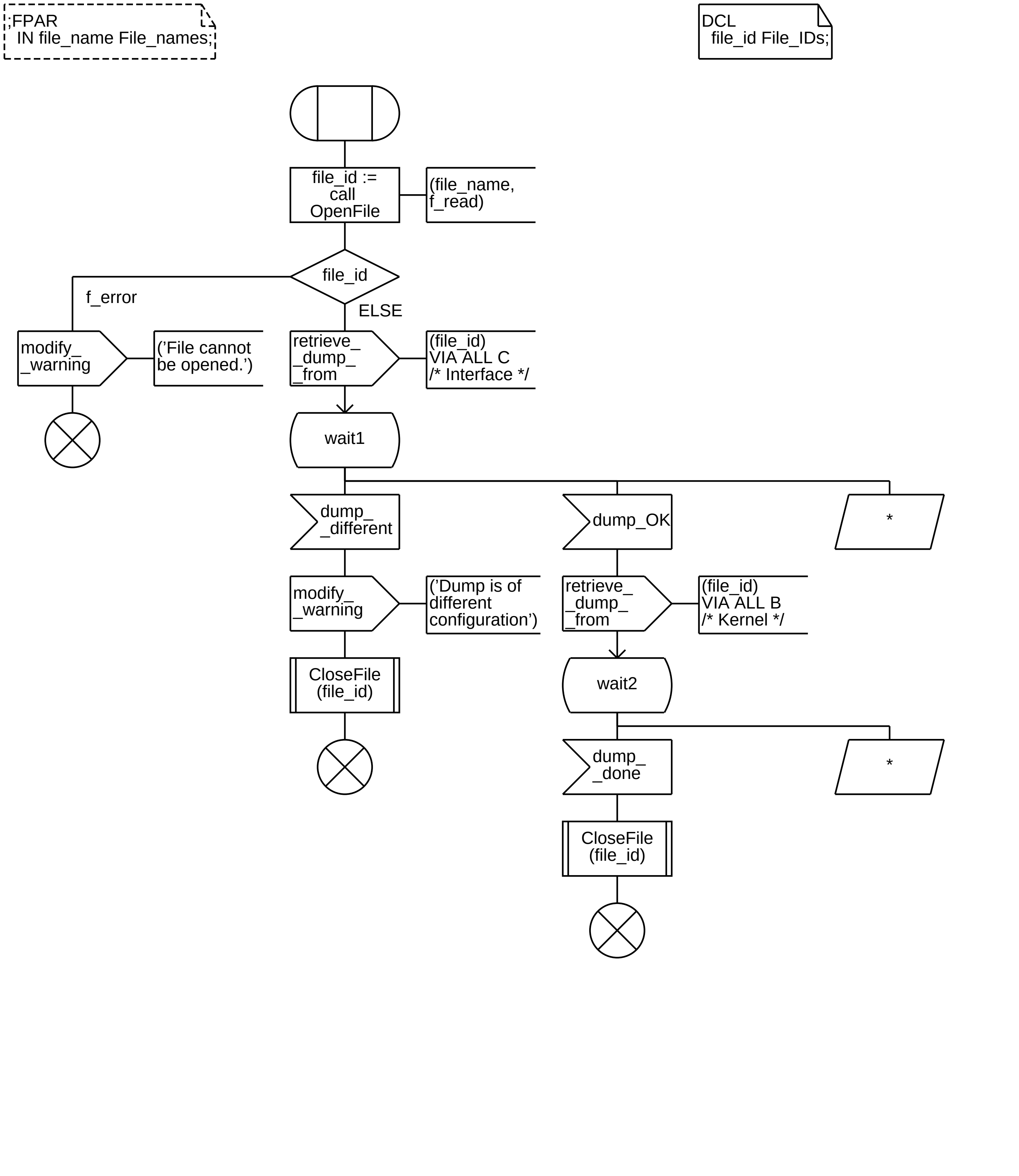

PROCEDURE GetID
1(1)
;
FPAR IN
name Names;
RETURNS
IDs;
DCL
returned_name Names;
id IDs;
get_id
(name)
wait
id
(id,
returned_name)
*
id
id_illegal
ELSE
returned_
_name
= name
TRUE
FALSE
simulation_
_warning
('Non-existing
object referenced')
simulation_
_warning
('Reference not
resolved')
id_illegal
id
id_illegal

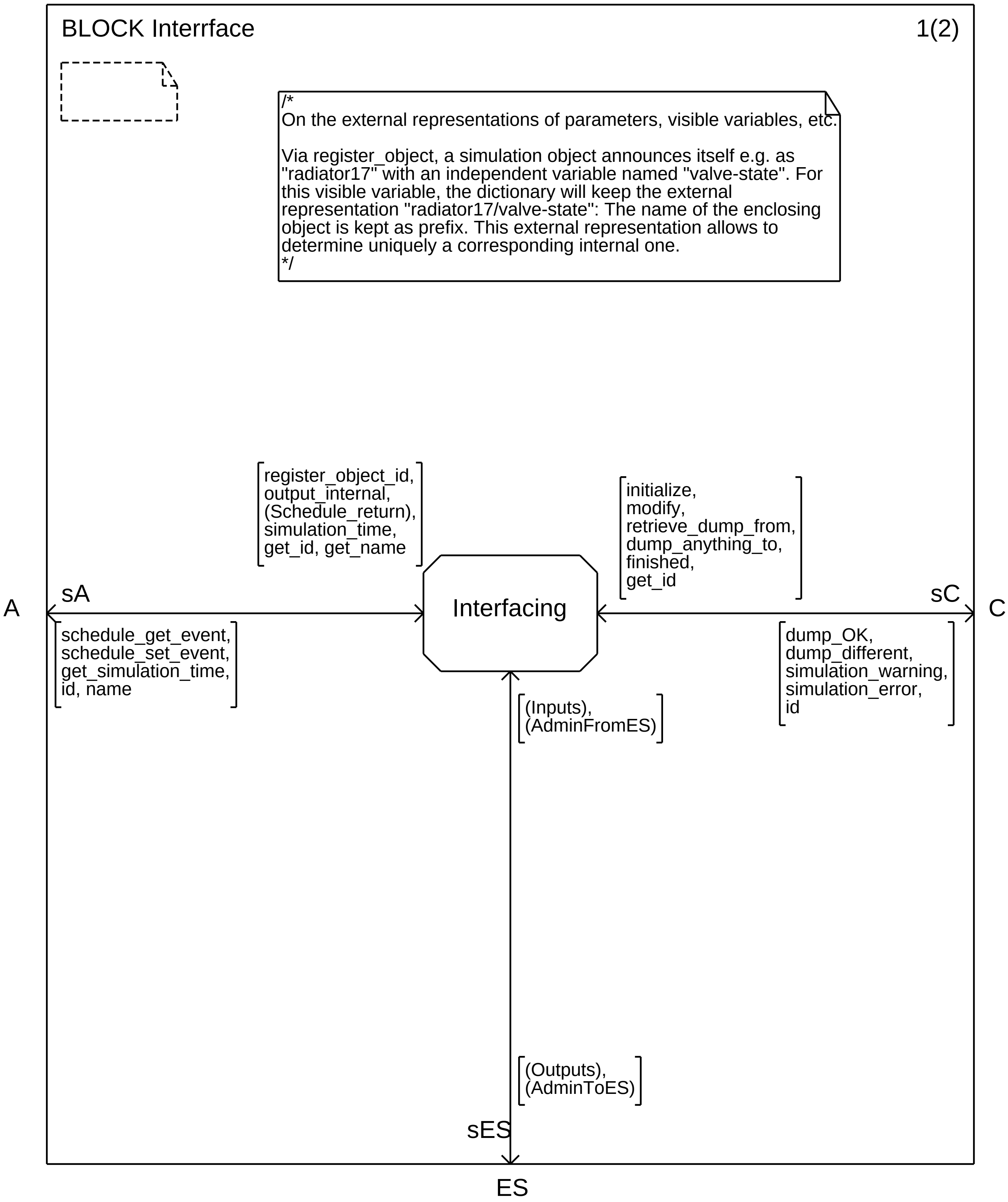
PROCEDURE <<PROCESS Control>> RetrieveDump


```
;FPAR
  IN file_name File_names;
```

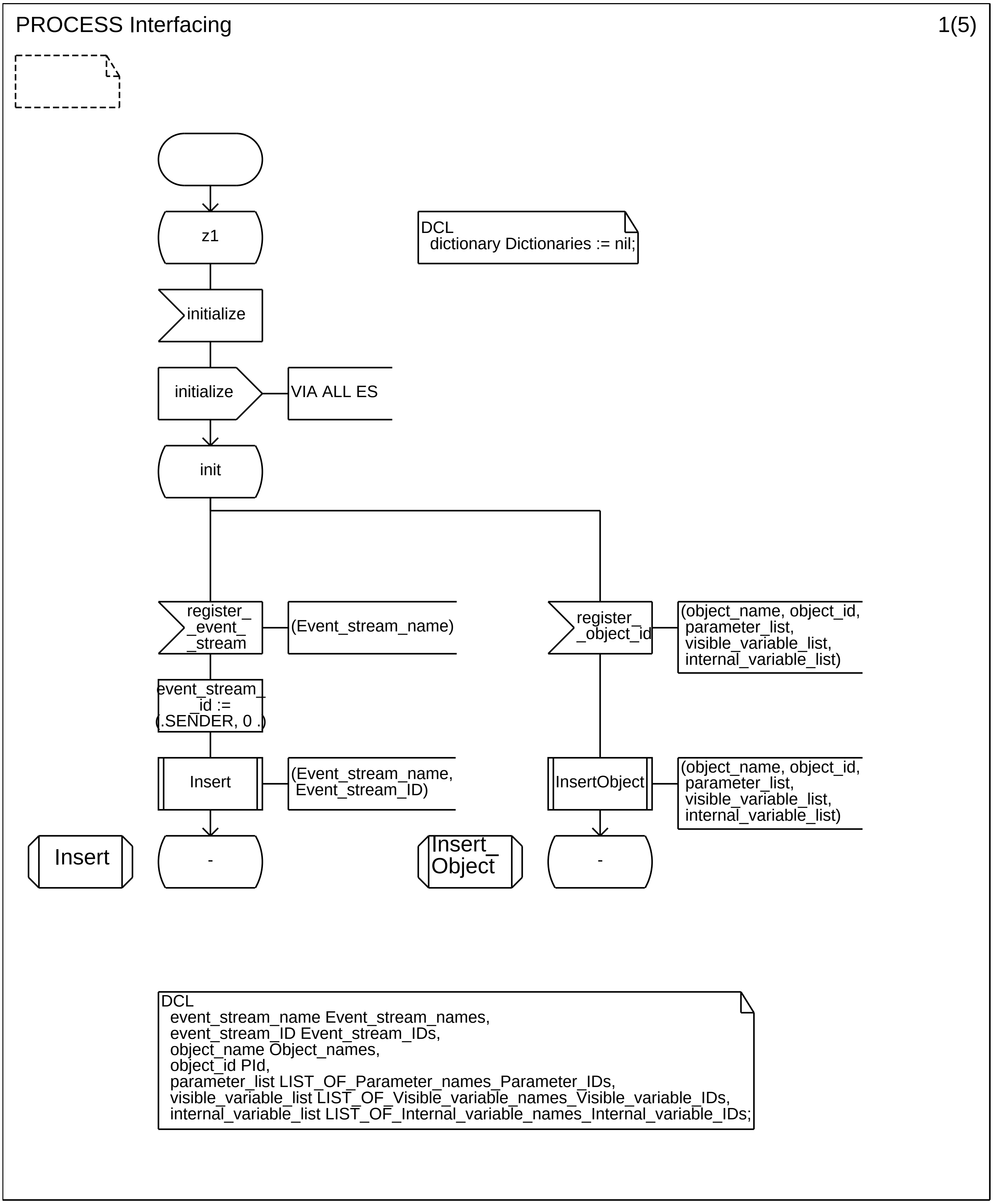
```
/*
On writing and retrieving dumps
==============================

(A) Complete dumps serving to re-start an identical run.

The dump is written into a text file (via calls to WriteString/Int/Real)
which is a sequence of strings that are separated by some delimiter.
The contents of this file is the following:

(1) *Sequence of all the external representations
    that are kept in the
    dictionary. External representations are just non-empty elements of
    Charstring. When retrieving, the simulator uses this sequence to check
    whether former and actual configuration are equal with respect to the
    simulation objects and the event streams.
    *An additional delimiter (= an empty string)
    denotes the end of the sequence.

(2) *Starting-point in simulation time (non-negative integer),
    *ending-point (ditto or "infinity"),
    *advancing-mode of simulation time
      either "fastest"
      or
      -factor (real),
      -slack (non-negative integer),
      -"cont"/"stop" (behaviour after exceeding slack),
    *sequence of halting-points, ended by additional delimiter

(3) Event collection:
    *sequence of vectors
      -execution time,
      -period ("0" in case of non-periodic events),
      -and
       (a) "get", Visible_variable_name, Event_stream_name for get events
       (b) "set", Visible_variable_name, value (as Charstring) for set events
       (c) "sim", Object_name for simulate  events
    *An additional delimiter (= an empty string)
    denotes the end of the sequence.

(4) State of objects as
    *sequence of
      (a) "par" parameter_name value
      (b) "vis" visible_variable_name value
      (c) "int" internal_variable_name value
      (d) "max" object_name duration
      (e) "out" visible_variable_name event_stream_name
      ended by an additional delimiter.

(B) Restricted dump or dump of visible variables

*simulation time
*sequence of
      (a) "par" parameter_name value
      (b) "vis" visible_variable_name value
 ended by an additional delimiter just before the file end.

*/
```

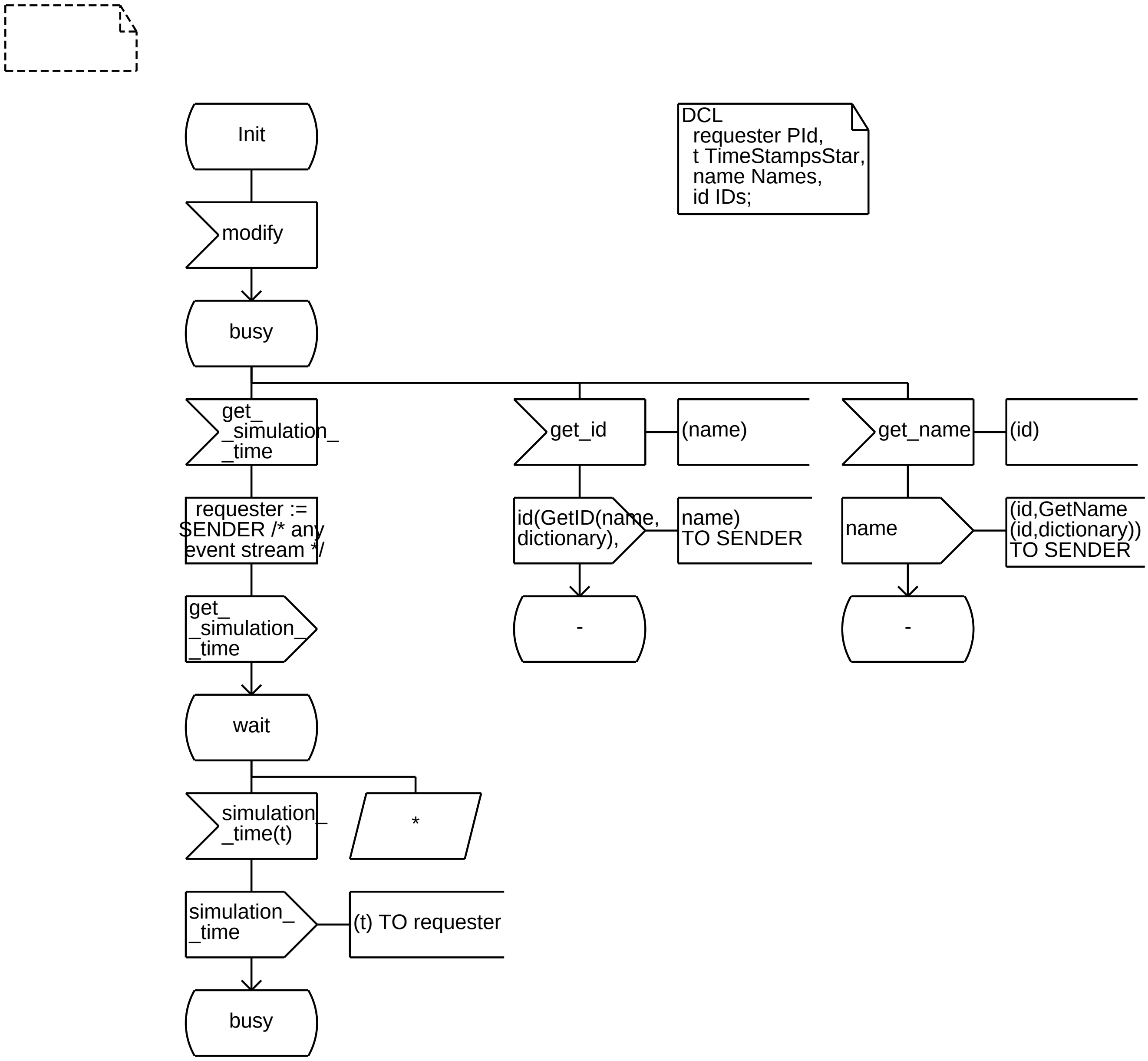
PROCEDURE <<PROCESS Control>> RetrieveDump

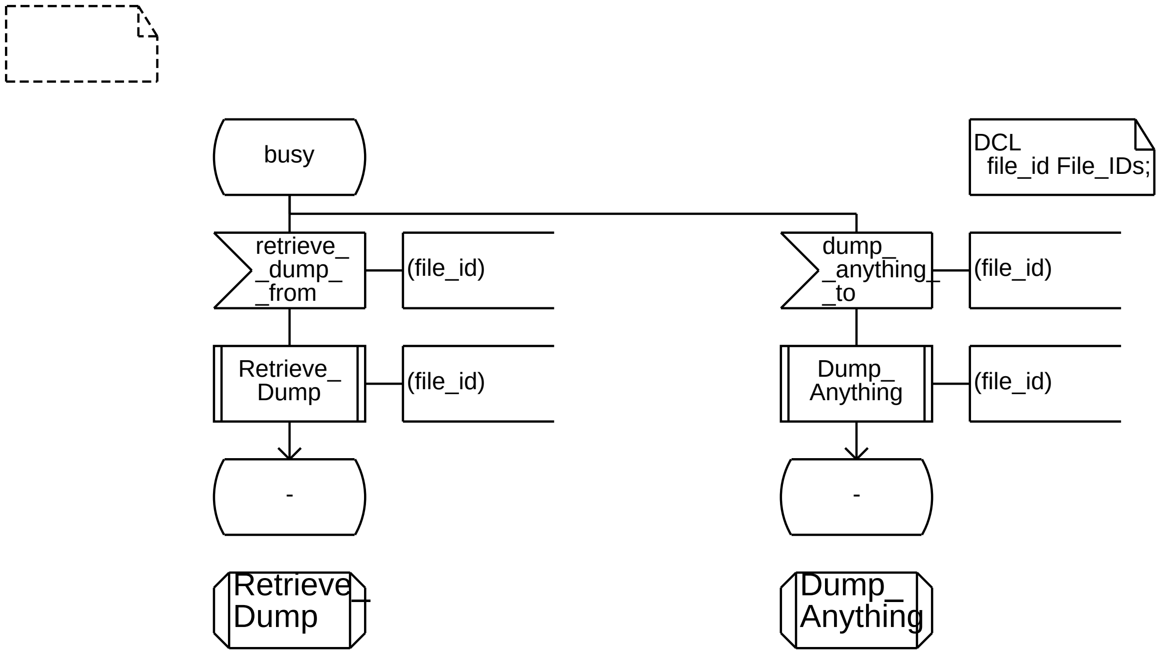


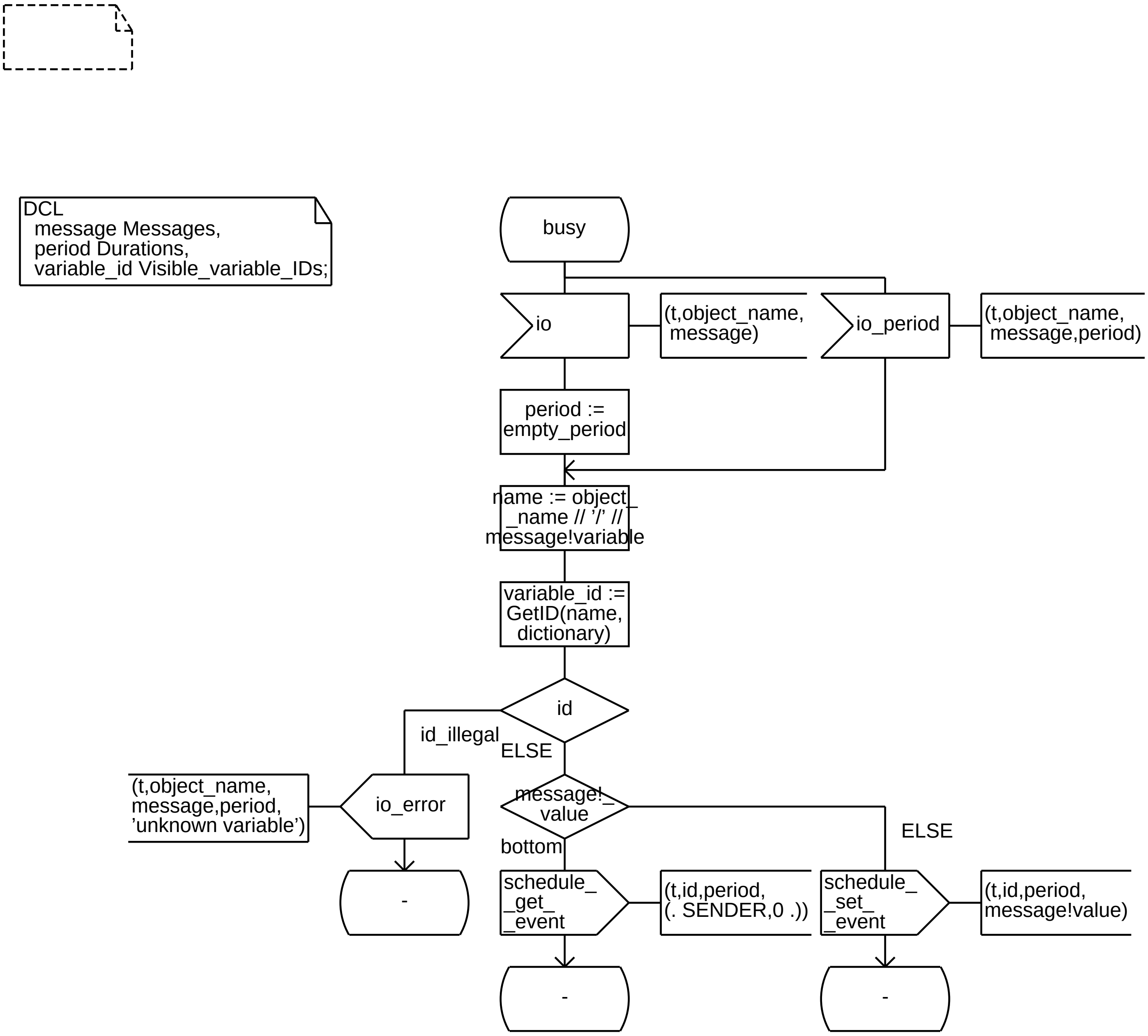

BLOCK Interrface
1(2)
/*
On the external representations of parameters, visible variables, etc.
Via register_object, a simulation object announces itself e.g. as "radiator17" with an independent variable named "valve-state". For this visible variable, the dictionary will keep the external representation "radiator17/valve-state": The name of the enclosing object is kept as prefix. This external representation allows to determine uniquely a corresponding internal one.
*/
register_object_id, output_internal, (Schedule_return), simulation_time, get_id, get_name
initialize, modify, retrieve_dump_from, dump_anything_to, finished, get_id
A
sA
Interfacing
sC
C
schedule_get_event, schedule_set_event, get_simulation_time, id, name
dump_OK, dump_different, simulation_warning, simulation_error, id
(Inputs), (AdminFromES)
(Outputs), (AdminToES)
sES
ES

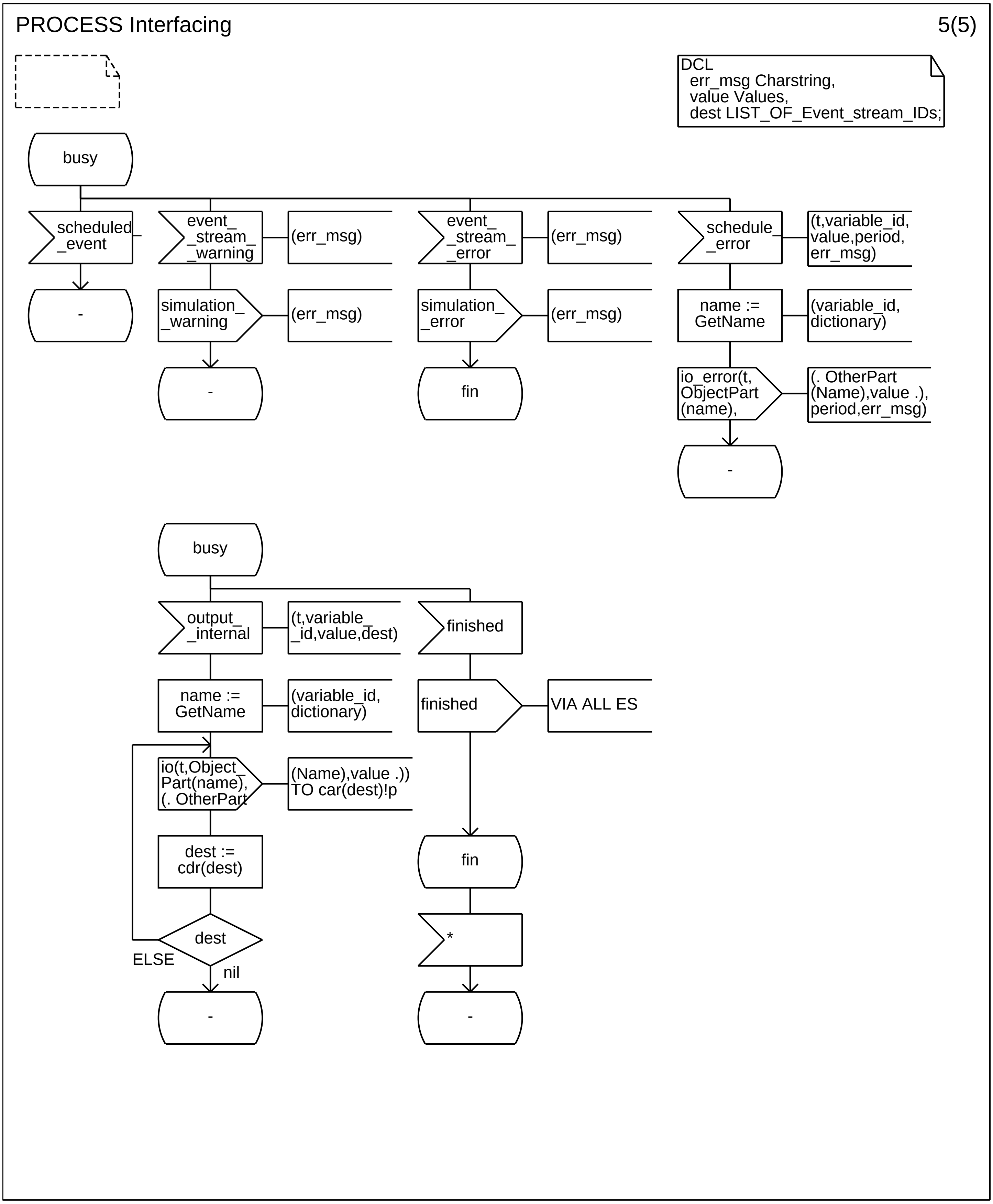
```
NEWTYPE DictionaryEntries
  STRUCT
    name Names;
    id IDs;
ENDNEWTYPE DictionaryEntries;


NEWTYPE Dictionaries
  List(DictionaryEntries)
ADDING
OPERATORS
  GetID:         Names,        Dictionaries -> IDs;
  GetName:       IDs,          Dictionaries -> Names;
  DictIsSubset:  Dictionaries, Dictionaries -> Boolean;
  DictEqual:     Dictionaries, Dictionaries -> Boolean;
AXIOMS
  FOR ALL d IN Dictionaries (
    FOR ALL e IN DictionaryEntries (
      FOR ALL n IN Names (
        GetID(n,nil) == id_illegal;
        GetID(n,cons(e,d)) ==
          IF n=nameEXTRACT!(e) THEN idEXTRACT!(e)
          ELSE GetID(n,d) FI;
      );
      FOR ALL id IN IDs (
        GetName(id,nil) == '';
        GetName(id,cons(e,d)) ==
          IF id=idEXTRACT!(e) THEN nameEXTRACT!(e)
          ELSE GetName(id,d) FI;
      );
      FOR ALL d2 IN Dictionaries (
        DictIsSubset(nil,d2)==TRUE;
        DictIsSubset(cons(e,d),d2) ==
          GetID(nameEXTRACT!(e),d2)/=id_illegal
          AND DictIsSubset(d,d2);
        DictEqual(d,d2) ==
          DictIsSubset(d,d2) AND DictIsSubset(d2,s);
      )
    )
  )
ENDNEWTYPE Dictionaries;
```


PROCESS Interfacing
1(5)
DCL
dictionary Dictionaries := nil;
z1
initialize
initialize
VIA ALL ES
init
register_ _event_ _stream
(Event_stream_name)
event_stream_ _id := (.SENDER, 0 .)
Insert
(Event_stream_name, Event_stream_ID)
Insert
-
register_ _object_id
(object_name, object_id, parameter_list, visible_variable_list, internal_variable_list)
InsertObject
(object_name, object_id, parameter_list, visible_variable_list, internal_variable_list)
Insert_ Object
-
DCL
event_stream_name Event_stream_names,
event_stream_ID Event_stream_IDs,
object_name Object_names,
object_id PId,
parameter_list LIST_OF_Parameter_names_Parameter_IDs,
visible_variable_list LIST_OF_Visible_variable_names_Visible_variable_IDs,
internal_variable_list LIST_OF_Internal_variable_names_Internal_variable_IDs;


DCL
requester PId,
t TimeStampsStar,
name Names,
id IDs;
Init
modify
busy
get_
simulation
_time
requester :=
SENDER /* any
event stream */
get_
simulation
_time
wait
simulation_
_time(t)
*
simulation_
_time
(t) TO requester
busy
get_id
(name)
id(GetID(name,
dictionary),
name)
TO SENDER
-
get_name
(id)
name
(id,GetName
(id,dictionary))
TO SENDER
-

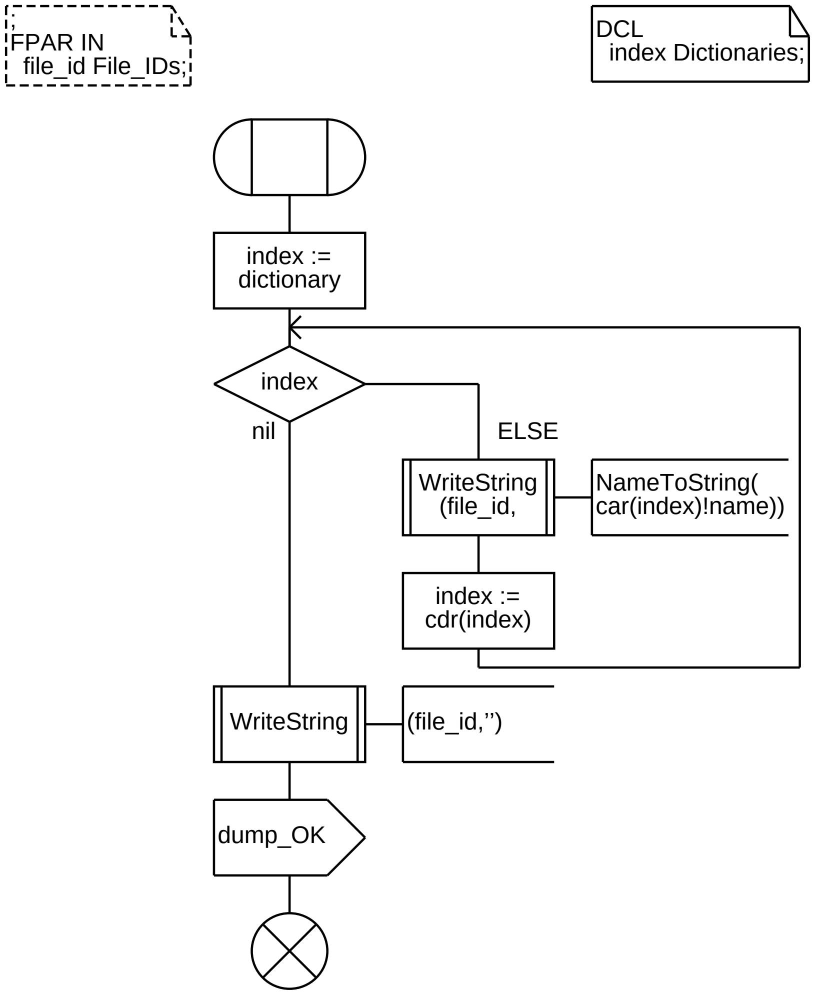

busy
DCL
file_id File_IDs;
retrieve_
dump
_from
(file_id)
dump_
anything
_to
(file_id)
Retrieve_
Dump
(file_id)
Dump_
Anything
(file_id)
-
-
Retrieve_
Dump
Dump_
Anything


DCL
message Messages,
period Durations,
variable_id Visible_variable_IDs;
busy
io
(t,object_name,
message)
io_period
(t,object_name,
message,period)
period :=
empty_period
name := object_
_name // '/' //
message!variable
variable_id :=
GetID(name,
dictionary)
id
id_illegal
ELSE
io_error
(t,object_name,
message,period,
'unknown variable')
-
message!_
value
bottom
ELSE
schedule_
get
_event
(t,id,period,
(. SENDER,0 .))
schedule_
set
_event
(t,id,period,
message!value)
-
-

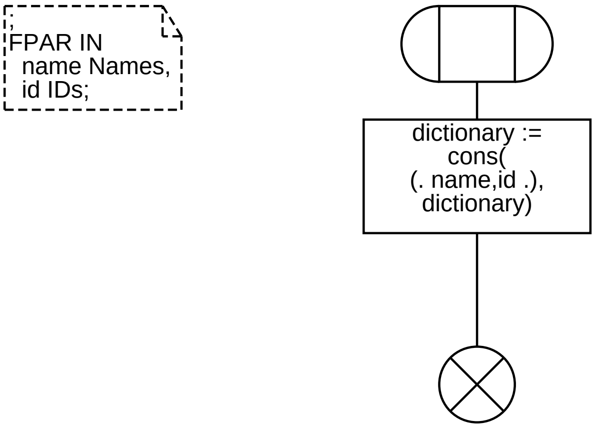


PROCESS Interfacing
5(5)
DCL
err_msg Charstring,
value Values,
dest LIST_OF_Event_stream_IDs;
busy
scheduled_
_event
-
event_
stream
_warning
(err_msg)
simulation_
_warning
(err_msg)
-
event_
stream
_error
(err_msg)
simulation_
_error
(err_msg)
fin
schedule_
_error
(t,variable_id,
value,period,
err_msg)
name :=
GetName
(variable_id,
dictionary)
io_error(t,
ObjectPart
(name),
(. OtherPart
(Name),value .),
period,err_msg)
-
busy
output_
_internal
(t,variable_
_id,value,dest)
name :=
GetName
(variable_id,
dictionary)
io(t,Object_
Part(name),
(. OtherPart
(Name),value .))
TO car(dest)!p
dest :=
cdr(dest)
dest
ELSE
nil
-
finished
finished
VIA ALL ES
fin
*
-

PROCEDURE <<PROCESS Interfacing>> DumpAnything 1(1)

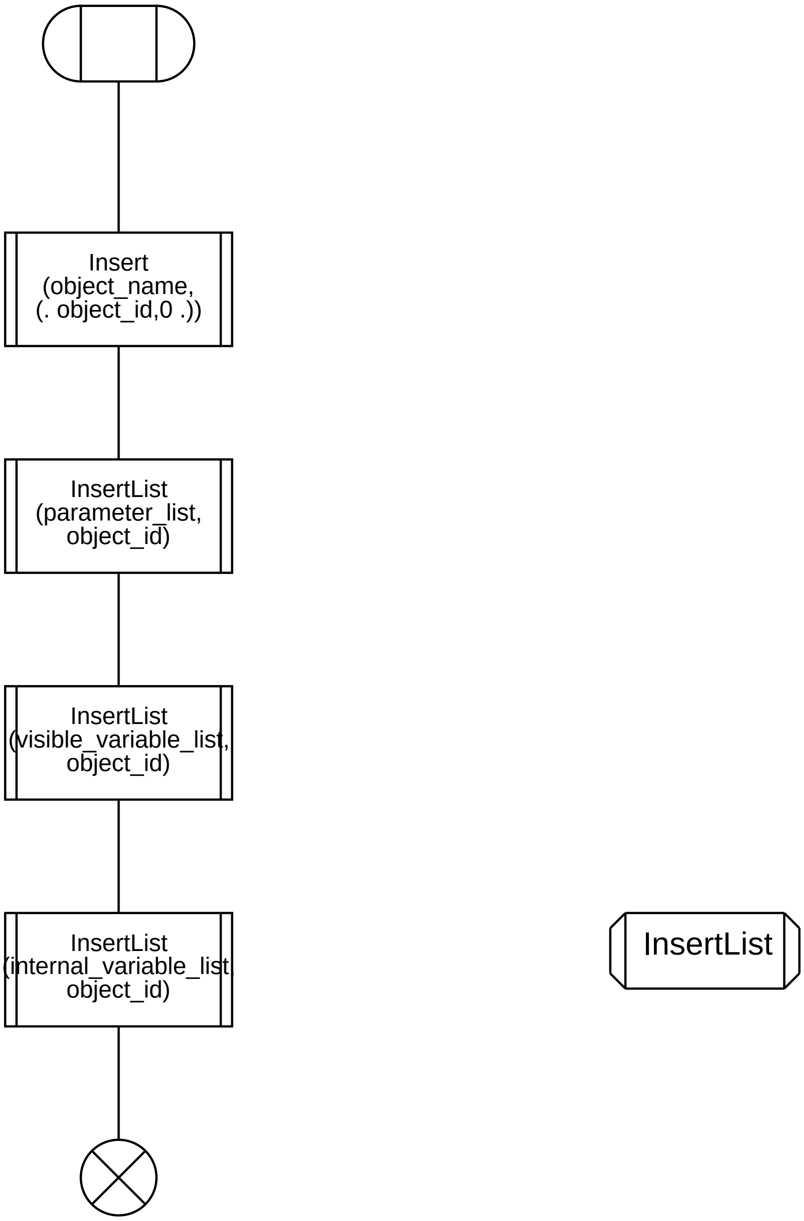


PROCEDURE Insert 1(1)

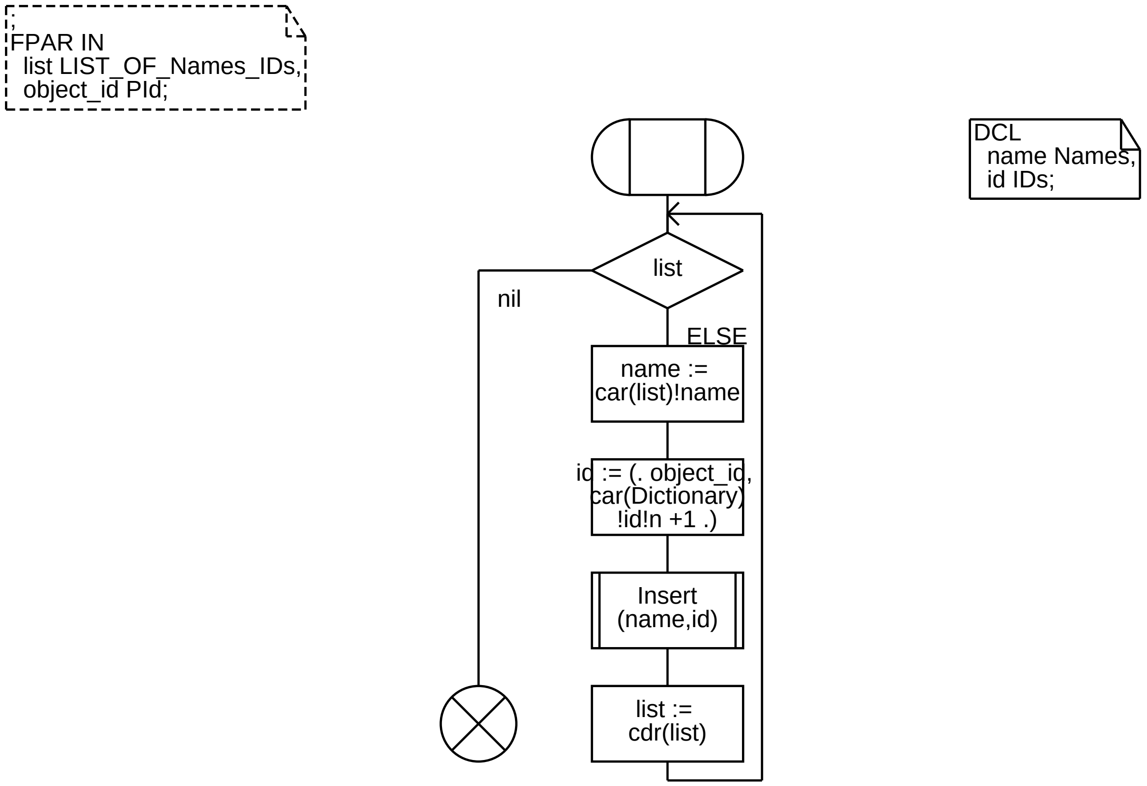


PROCEDURE InsertObject 1(1)

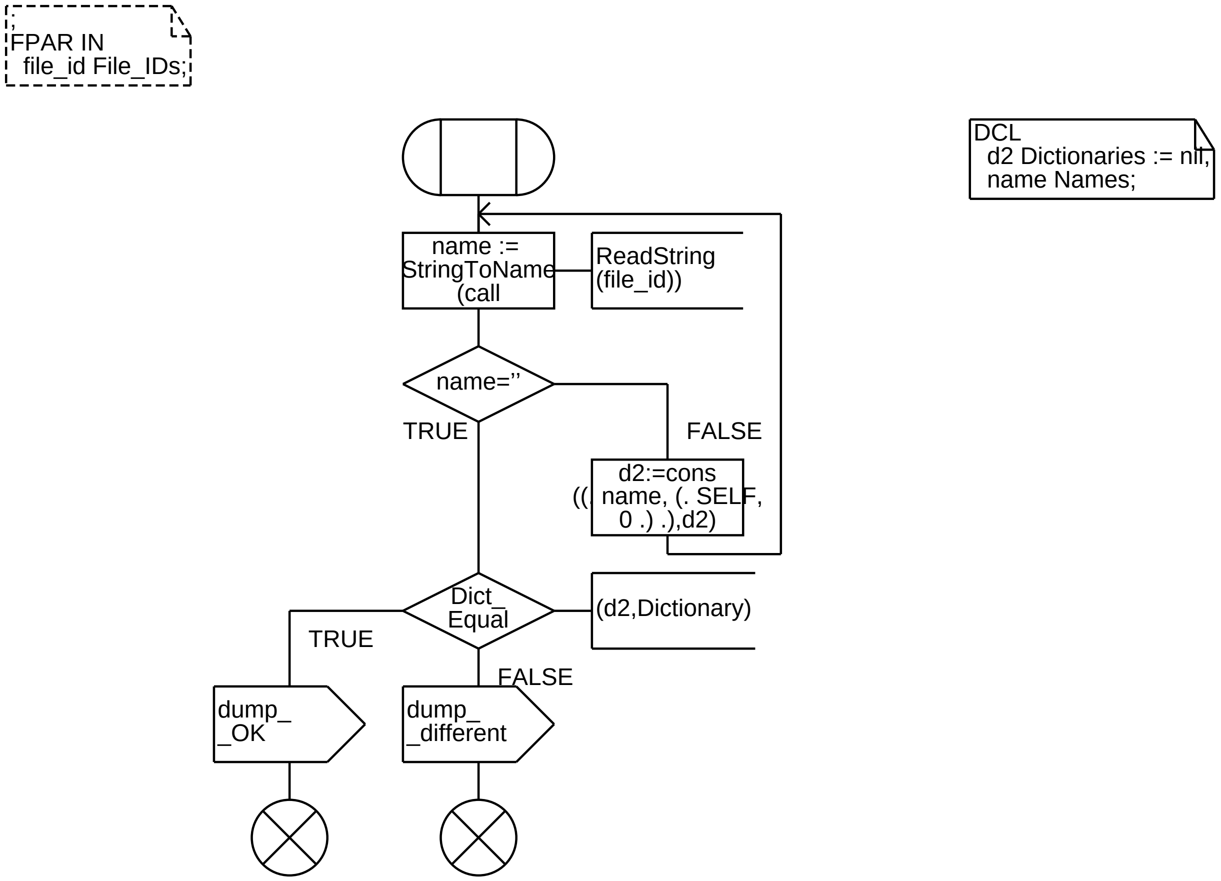
```
;
FPAR IN
  object_name Object_names,
  object_id PId,
  parameter_list LIST_OF_Parameter_names_Parameter_IDs,
  visible_variable_list LIST_OF_Visible_variable_names_Visible_variable_IDs,
  internal_variable_list LIST_OF_Internal_variable_names_Internal_variable_IDs;
```

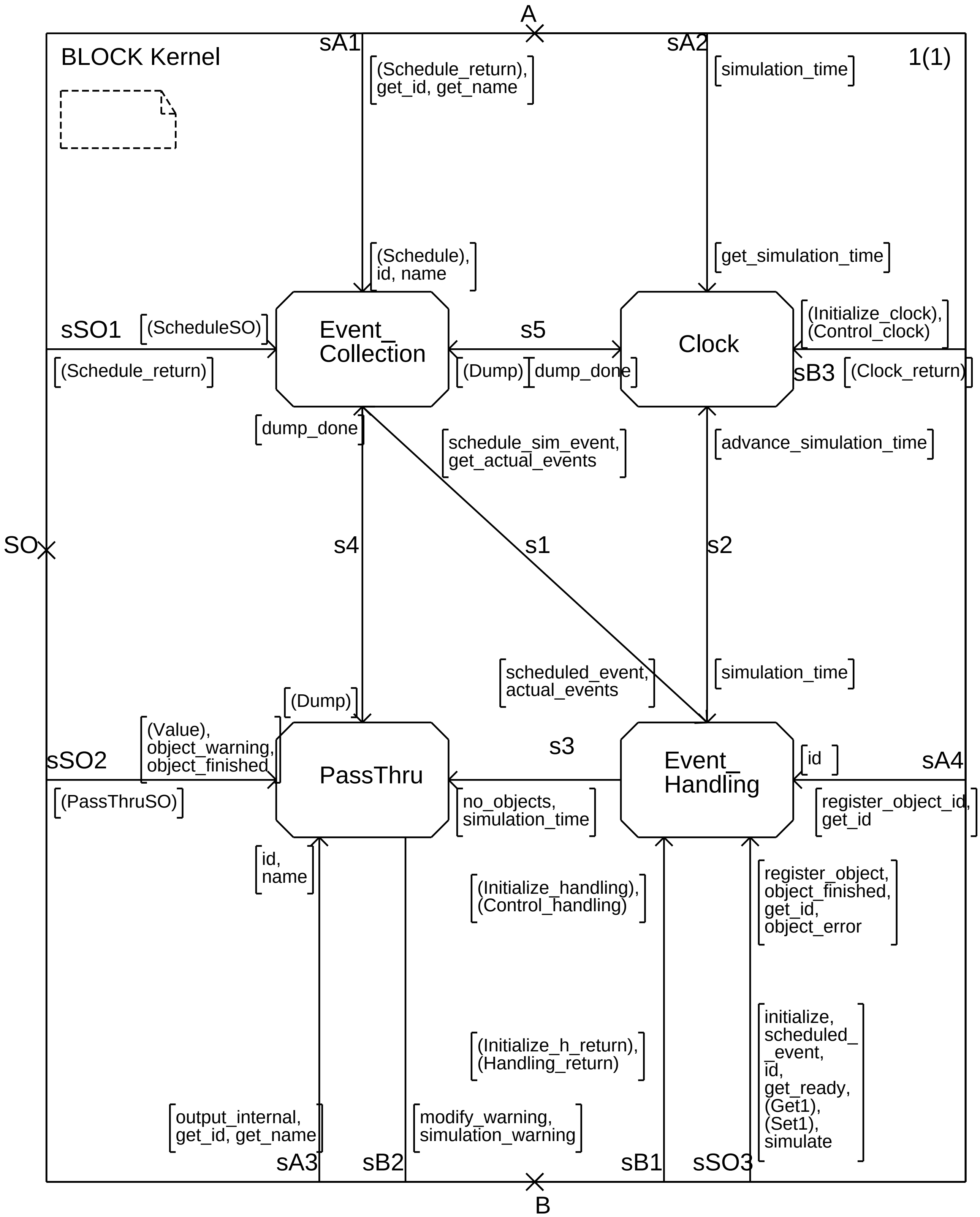


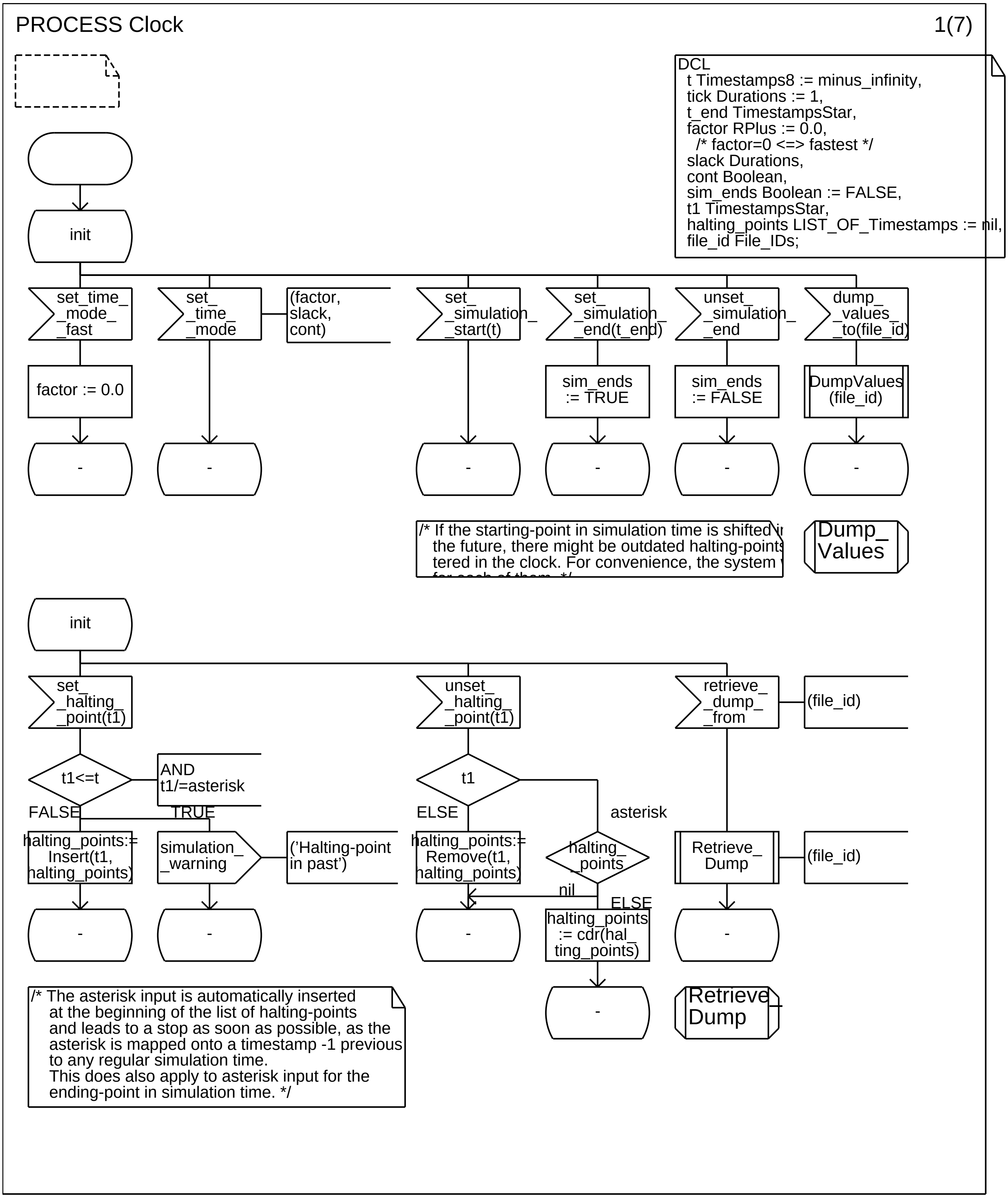
PROCEDURE InsertList 1(1)



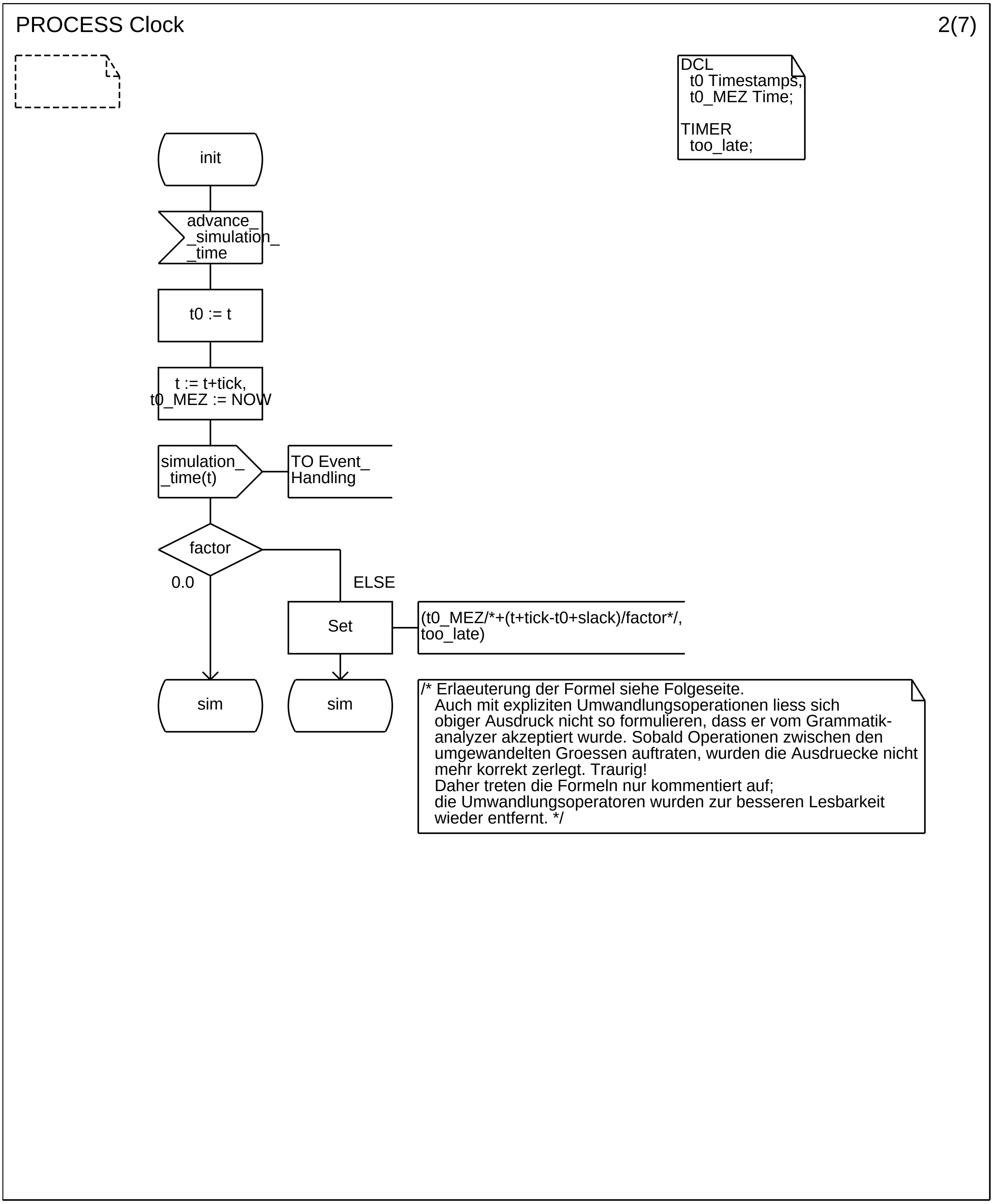
PROCEDURE <<PROCESS Interfacing>> RetrieveDump 1(1)



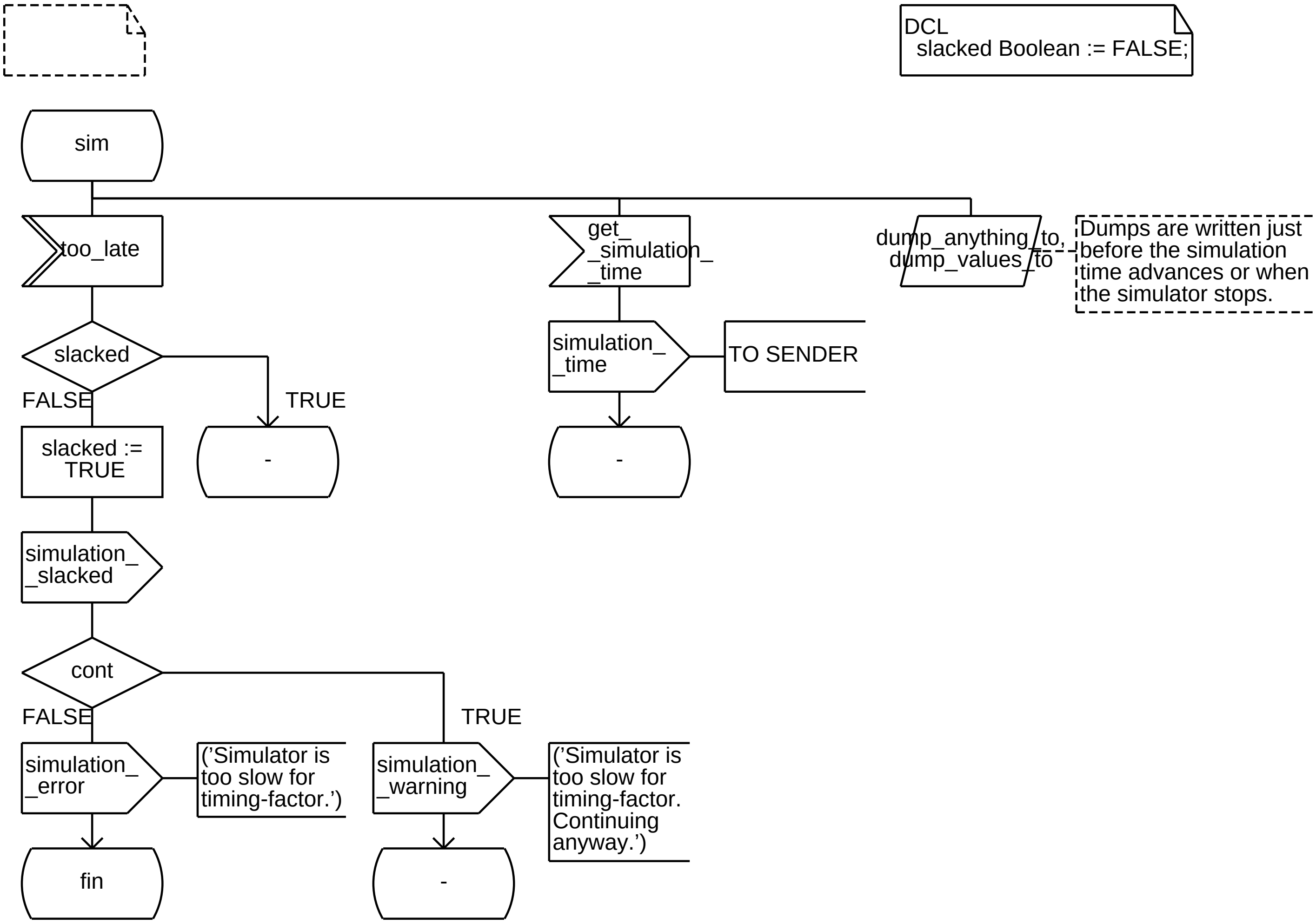

A
BLOCK Kernel
1(1)
sA1
sA2
(Schedule_return), get_id, get_name
simulation_time
(Schedule), id, name
get_simulation_time
Event_ Collection
Clock
sSO1
(ScheduleSO)
(Schedule_return)
s5
(Dump)
dump_done
(Initialize_clock), (Control_clock)
sB3
(Clock_return)
dump_done
schedule_sim_event, get_actual_events
advance_simulation_time
SO
s4
s1
s2
scheduled_event, actual_events
simulation_time
(Dump)
(Value), object_warning, object_finished
sSO2
PassThru
s3
Event_ Handling
id
sA4
(PassThruSO)
no_objects, simulation_time
register_object_id, get_id
id, name
(Initialize_handling), (Control_handling)
register_object, object_finished, get_id, object_error
initialize, scheduled_ _event, id, get_ready, (Get1), (Set1), simulate
(Initialize_h_return), (Handling_return)
output_internal, get_id, get_name
modify_warning, simulation_warning
sA3
sB2
sB1
sSO3
B

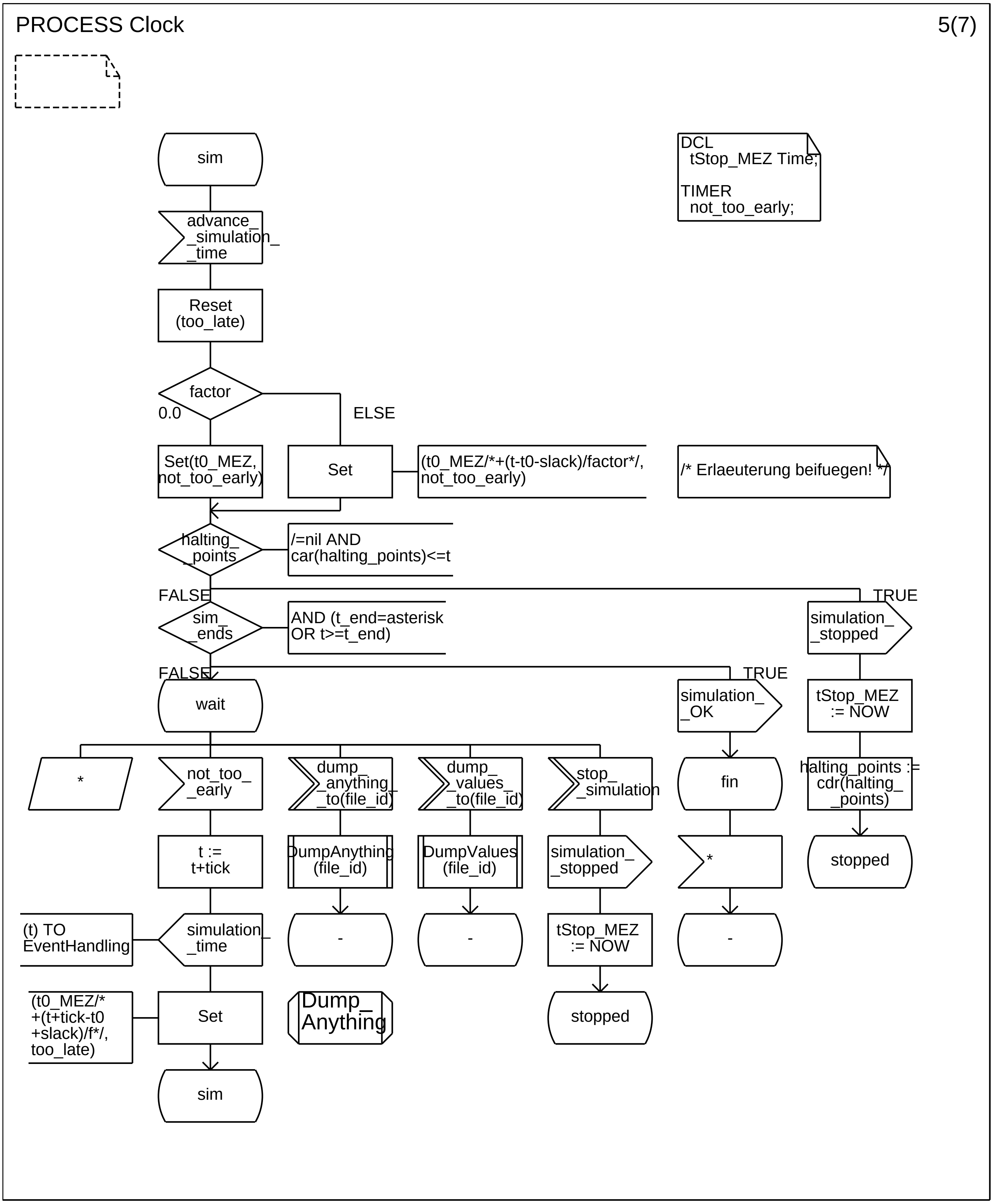

PROCESS Clock
1(7)
DCL
t Timestamps8 := minus_infinity,
tick Durations := 1,
t_end TimestampsStar,
factor RPlus := 0.0,
/* factor=0 <=> fastest */
slack Durations,
cont Boolean,
sim_ends Boolean := FALSE,
t1 TimestampsStar,
halting_points LIST_OF_Timestamps := nil,
file_id File_IDs;
init
set_time_ _mode_ _fast
factor := 0.0
set_ _time_ _mode
(factor, slack, cont)
set_ _simulation_ _start(t)
set_ _simulation_ _end(t_end)
sim_ends := TRUE
unset_ _simulation_ _end
sim_ends := FALSE
dump_ _values_ _to(file_id)
DumpValues (file_id)
-
/* If the starting-point in simulation time is shifted i the future, there might be outdated halting-points tered in the clock. For convenience, the system
Dump_ Values
init
set_ _halting_ _point(t1)
t1<=t
AND t1/=asterisk
FALSE
TRUE
halting_points:= Insert(t1, halting_points)
simulation_ _warning
('Halting-point in past')
unset_ _halting_ _point(t1)
t1
ELSE
asterisk
halting_points:= Remove(t1, halting_points)
halting_ _points
nil
ELSE
halting_points := cdr(hal_ ting_points)
retrieve_ _dump_ _from
(file_id)
Retrieve_ Dump
(file_id)
Retrieve_ Dump
/* The asterisk input is automatically inserted at the beginning of the list of halting-points and leads to a stop as soon as possible, as the asterisk is mapped onto a timestamp -1 previous to any regular simulation time.
This does also apply to asterisk input for the ending-point in simulation time. */

```
/* If the starting-point in simulation time is shifted into
   the future, there might be outdated halting-points regis-
   tered in the clock. For convenience, the system will stop
   for each of them. */
```

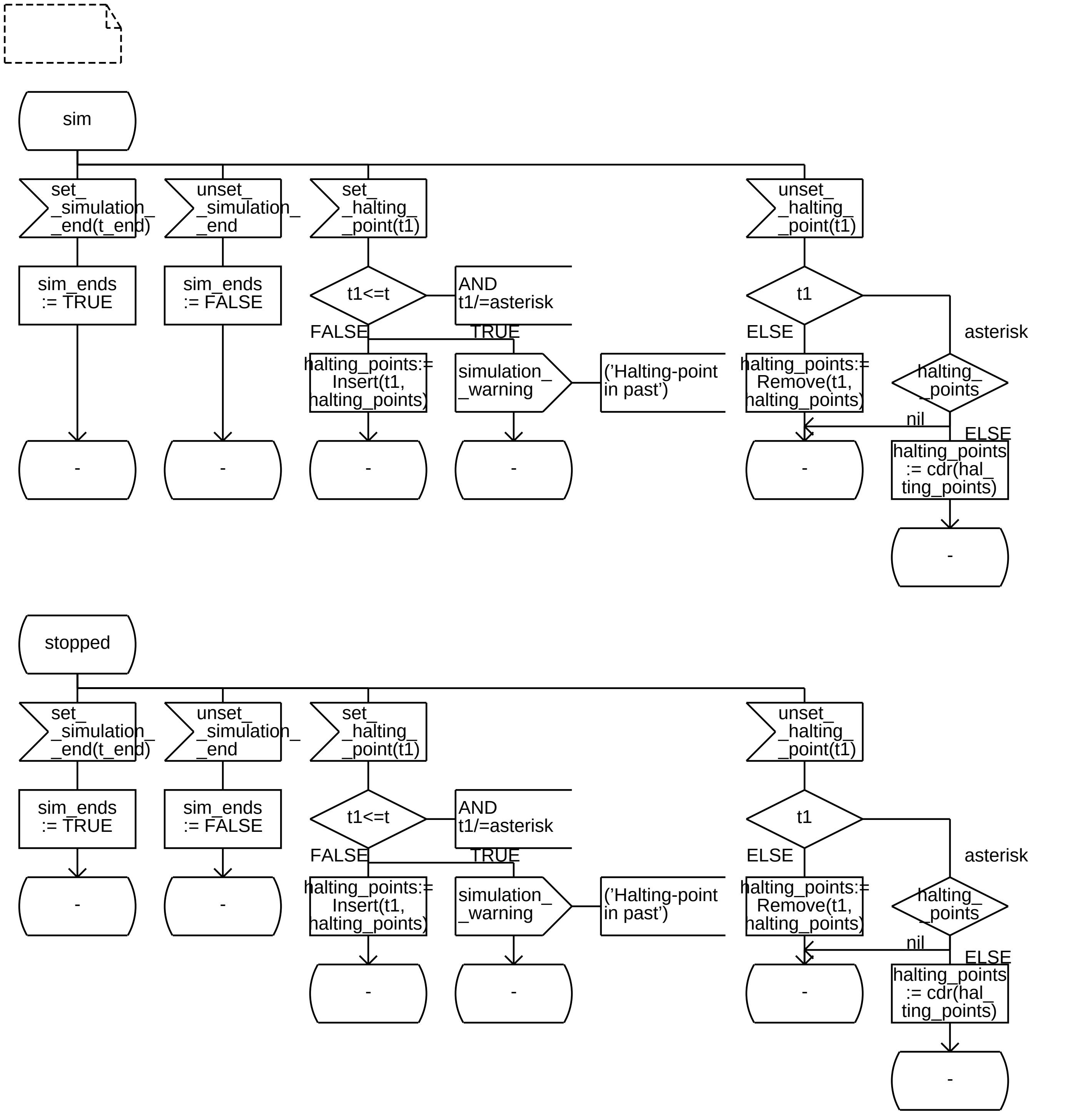

PROCESS Clock
2(7)
DCL
t0 Timestamps,
t0_MEZ Time;
TIMER
too_late;
init
advance_
simulation
_time
t0 := t
t := t+tick,
t0_MEZ := NOW
simulation_
_time(t)
TO Event_
Handling
factor
0.0
ELSE
Set
(t0_MEZ/*+(t+tick-t0+slack)/factor*/,
too_late)
sim
sim
/* Erlaeuterung der Formel siehe Folgeseite.
Auch mit expliziten Umwandlungsoperationen liess sich
obiger Ausdruck nicht so formulieren, dass er vom Grammatik-
analyzer akzeptiert wurde. Sobald Operationen zwischen den
umgewandelten Groessen auftraten, wurden die Ausdruecke nicht
mehr korrekt zerlegt. Traurig!
Daher treten die Formeln nur kommentiert auf;
die Umwandlungsoperatoren wurden zur besseren Lesbarkeit
wieder entfernt. */

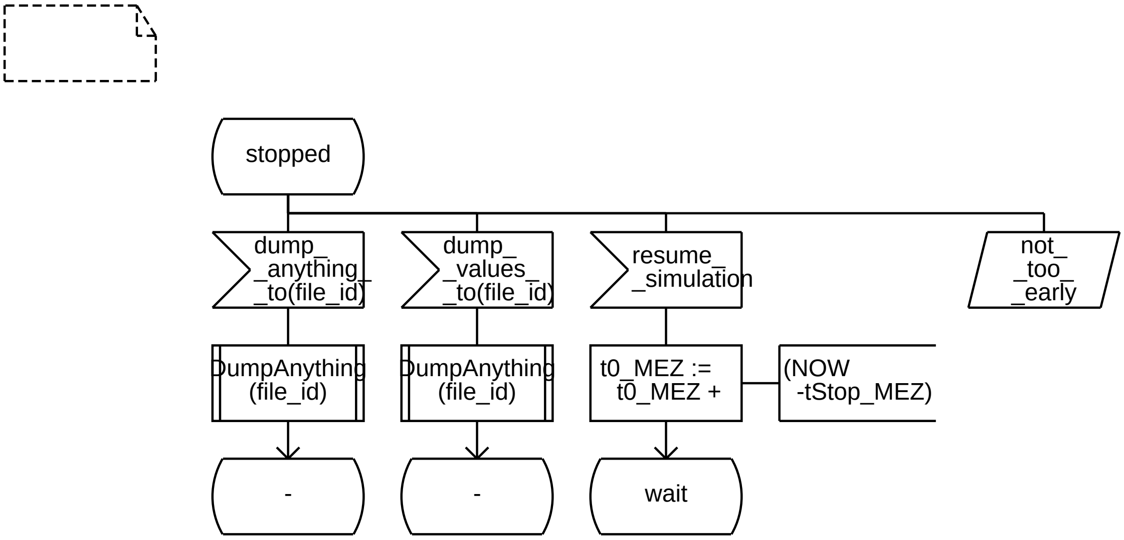
```
Realisierung der Kopplung Simulationszeit - Echtzeit

Verwendete Bezeichner:
======================
----------------+----------------------+--------------
 t0 Timestamps  |  Simulationsstart    |  t0_MEZ TIME
----------------+----------------------+--------------
 t  ----"-----  |  beliebig spaeter    |  t_MEZ  -"--
----------------+----------------------+--------------
 t' ----"-----  |  unmittelbare,       |  t'_MEZ -"--
                |  diskrete Folgezeit  |
----------------+----------------------+--------------
----------------+-------------------------------------
 f  RPlus       |  Timing-Factor: 1 Echtzeit = f SZeit
----------------+-------------------------------------
 s  Durations   |  Schlupf
----------------+-------------------------------------
 d  Durations   |  Zeitaufloesung
----------------+-------------------------------------


I.   Welche Simulationszeit t muss bei Echtzeit t_MEZ
     erreicht worden sein?

     t  !=  (t_MEZ - t0_MEZ) * f + t0


II.  Erlaubte Abweichung:

     (t_MEZ - t0_MEZ) * f + t0 - s  <=  t  <=  (t_MEZ - t0_MEZ) * f + t0 + s


III. Bedeutung von Abweichungen:

     Simulator zu langsam  <=>  t > (t_MEZ - t0_MEZ) * f + t0 + s

     Simulator zu schnell  <=>  t < (t_MEZ - t0_MEZ) * f + t0 - s


IV.  Uebertragung in MEZ:

     Simulator zu langsam  <=>  t_MEZ < (t - t0 - s) / f + t0_MEZ

     Simulator zu schnell  <=>  t_MEZ > (t - t0 + s) / f + t0_MEZ


V.   Timer fuer naechsten Tick:

     Simulator nach naechstem Tick zu langsam

     <=> t + d  <  (t'_MEZ - t0_MEZ) * f + t0 - s

     <=> t'_MEZ  >  (t + d - t0 + s) / f + t0_MEZ
```

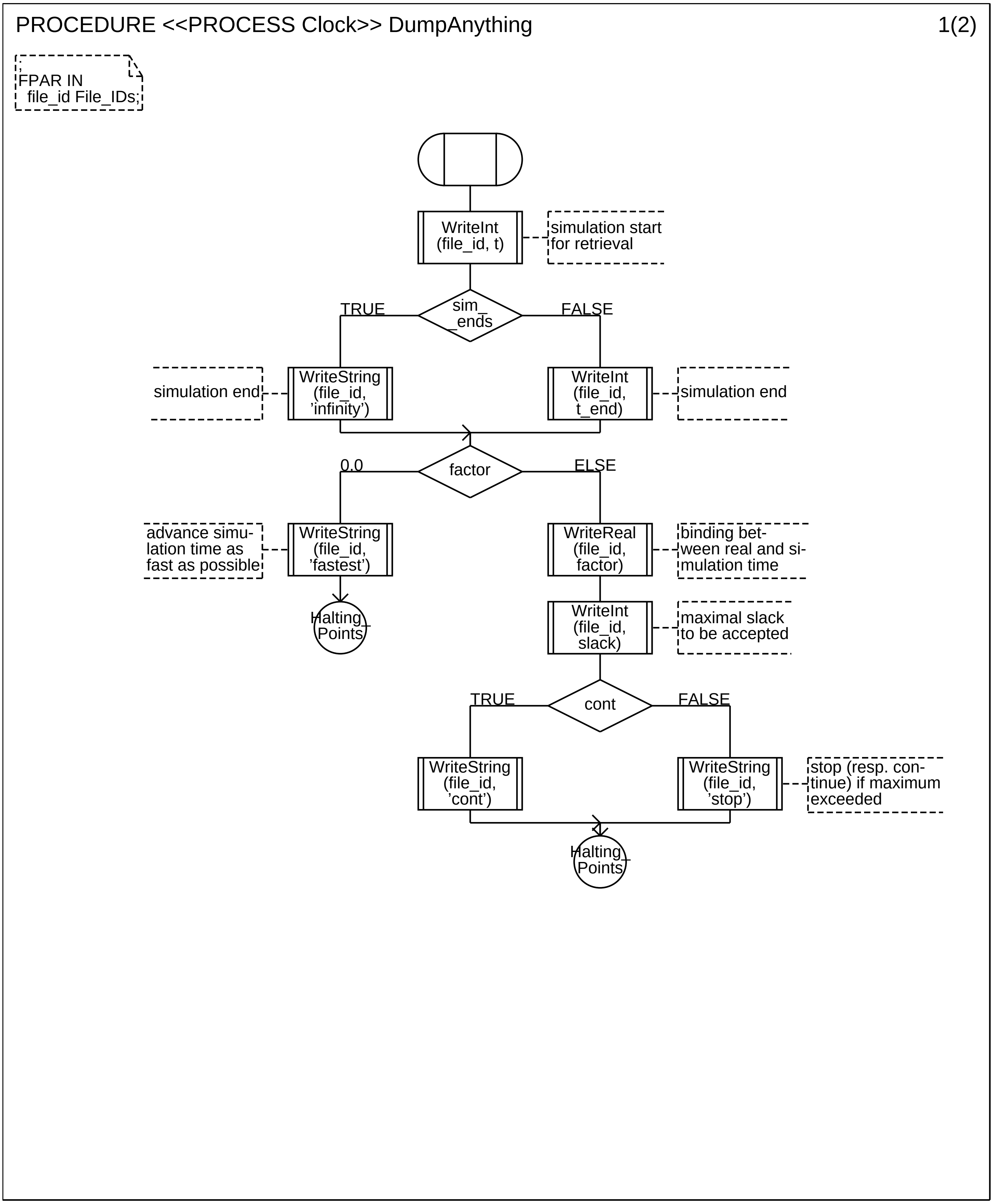

DCL
slacked Boolean := FALSE;
sim
too_late
get_
simulation
_time
dump_anything_to,
dump_values_to
Dumps are written just before the simulation time advances or when the simulator stops.
slacked
FALSE
TRUE
slacked := TRUE
-
simulation_
_time
TO SENDER
-
simulation_
_slacked
cont
FALSE
TRUE
simulation_
_error
('Simulator is too slow for timing-factor.')
simulation_
_warning
('Simulator is too slow for timing-factor. Continuing anyway.')
fin
-

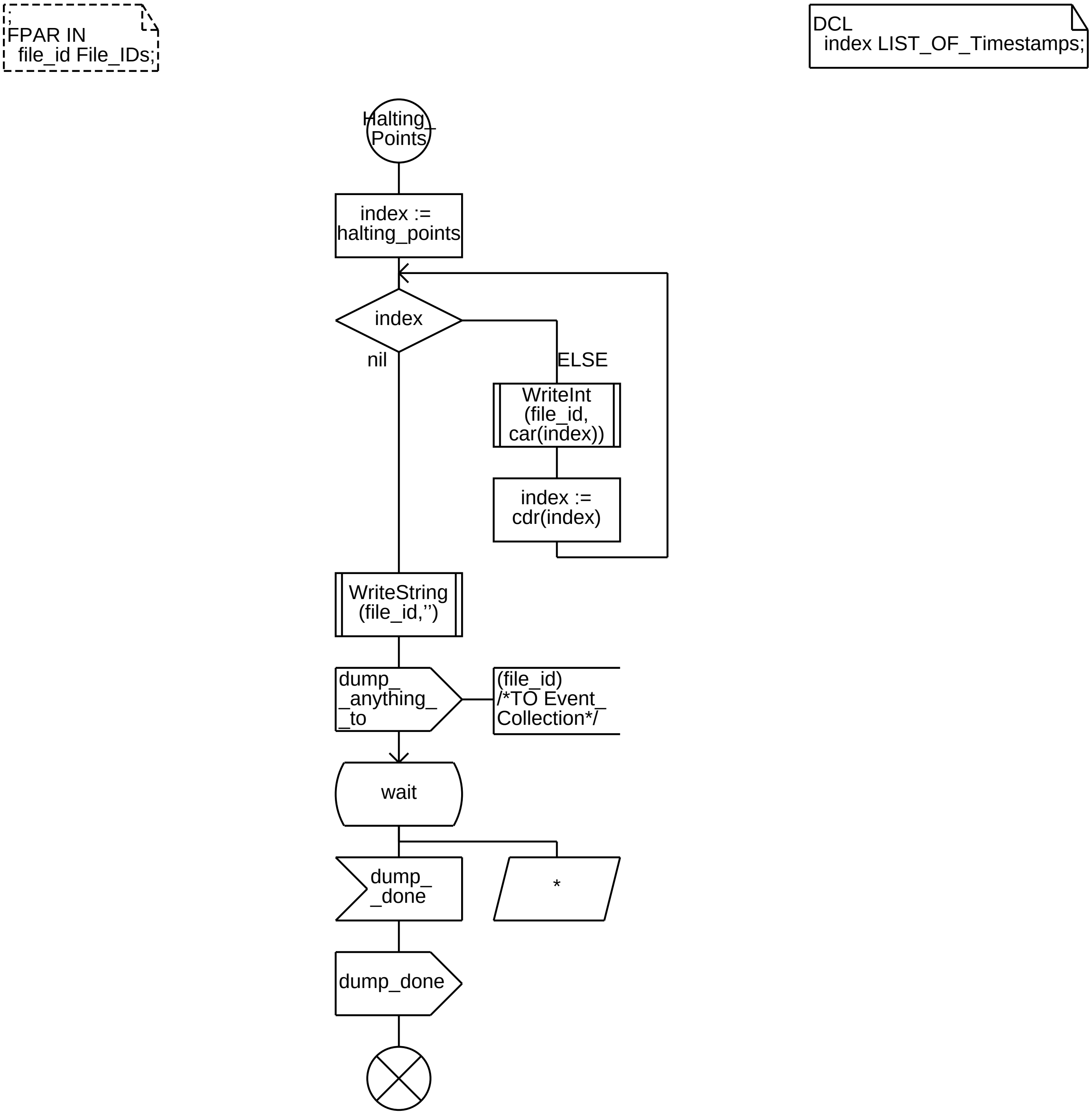

PROCESS Clock
5(7)
DCL
tStop_MEZ Time;
TIMER
not_too_early;
sim
advance_
simulation
_time
Reset
(too_late)
factor
0.0
ELSE
Set(t0_MEZ,
not_too_early)
Set
(t0_MEZ/*+(t-t0-slack)/factor*/,
not_too_early)
/* Erlaeuterung beifuegen! */
halting_
_points
/=nil AND
car(halting_points)<=t
FALSE
TRUE
sim_
_ends
AND (t_end=asterisk
OR t>=t_end)
simulation_
_stopped
FALSE
TRUE
wait
simulation_
_OK
tStop_MEZ
:= NOW
*
not_too_
_early
dump_
anything
_to(file_id)
dump_
values
_to(file_id)
stop_
_simulation
fin
halting_points :=
cdr(halting_
_points)
t :=
t+tick
DumpAnything
(file_id)
DumpValues
(file_id)
simulation_
_stopped
*
stopped
(t) TO
EventHandling
simulation_
_time
-
-
tStop_MEZ
:= NOW
-
(t0_MEZ/*
+(t+tick-t0
+slack)/f*/,
too_late)
Set
Dump_
Anything
stopped
sim

PROCESS Clock

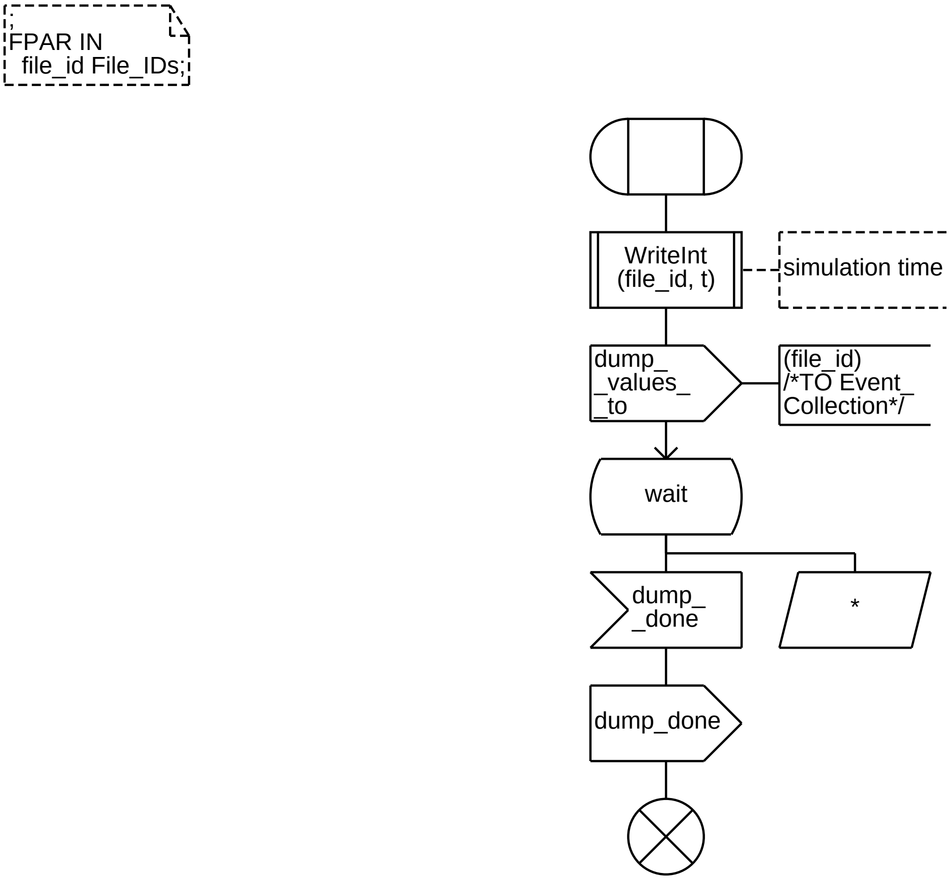



stopped
dump_ _anything_ _to(file_id)
dump_ _values_ _to(file_id)
resume_ _simulation
not_ _too_ _early
DumpAnything (file_id)
DumpAnything (file_id)
t0_MEZ := t0_MEZ +
(NOW -tStop_MEZ)
-
-
wait

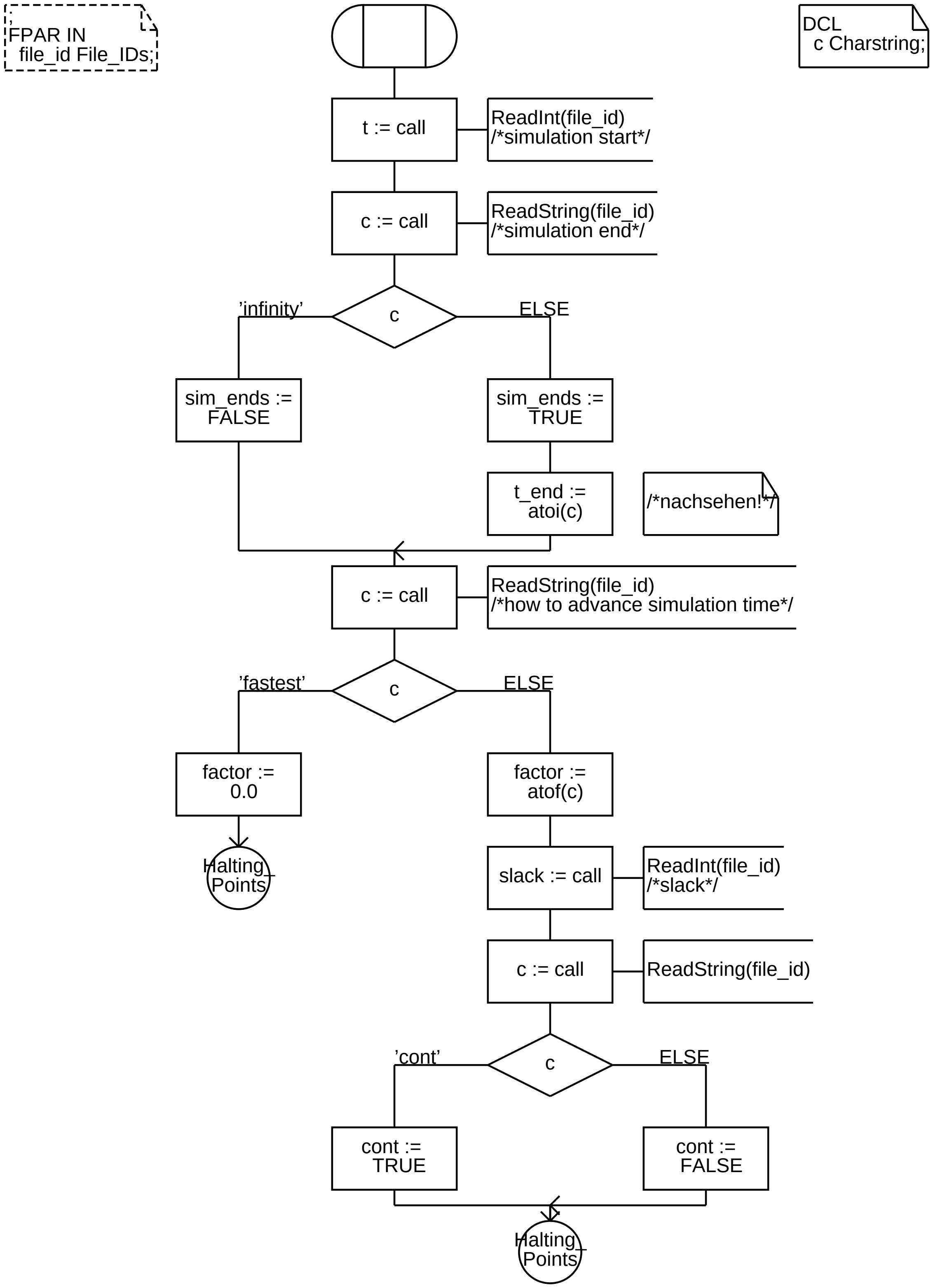

PROCEDURE <<PROCESS Clock>> DumpAnything
1(2)
;
FPAR IN
file_id File_IDs;
WriteInt
(file_id, t)
simulation start
for retrieval
sim_
_ends
TRUE
FALSE
simulation end
WriteString
(file_id,
'infinity')
WriteInt
(file_id,
t_end)
simulation end
factor
0.0
ELSE
advance simu-
lation time as
fast as possible
WriteString
(file_id,
'fastest')
WriteReal
(file_id,
factor)
binding bet-
ween real and si-
mulation time
Halting_
Points
WriteInt
(file_id,
slack)
maximal slack
to be accepted
cont
TRUE
FALSE
WriteString
(file_id,
'cont')
WriteString
(file_id,
'stop')
stop (resp. con-
tinue) if maximum
exceeded
Halting_
Points


;
FPAR IN
file_id File_IDs;
DCL
index LIST_OF_Timestamps;
Halting_
Points
index :=
halting_points
index
nil
ELSE
WriteInt
(file_id,
car(index))
index :=
cdr(index)
WriteString
(file_id,'')
dump_
anything
_to
(file_id)
/*TO Event_
Collection*/
wait
dump_
_done
*
dump_done

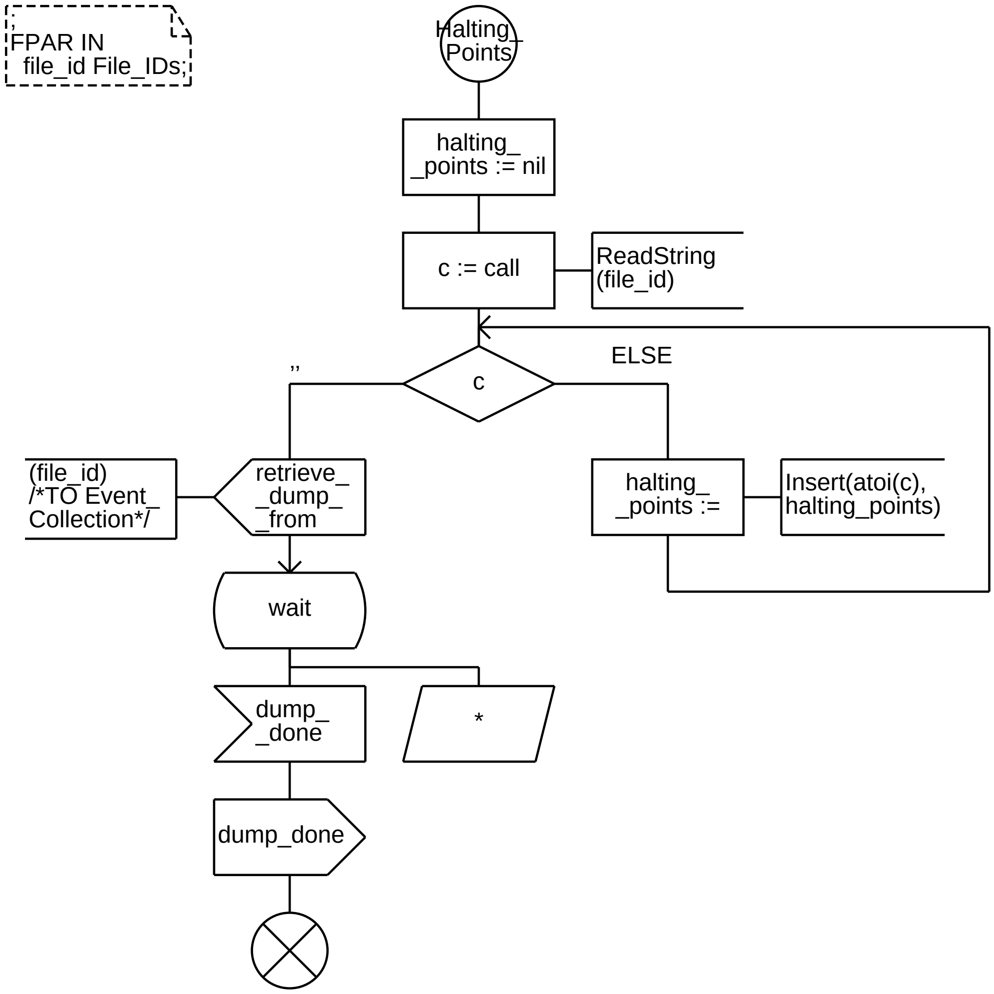
PROCEDURE <<PROCESS Clock>> DumpValues 1(1)



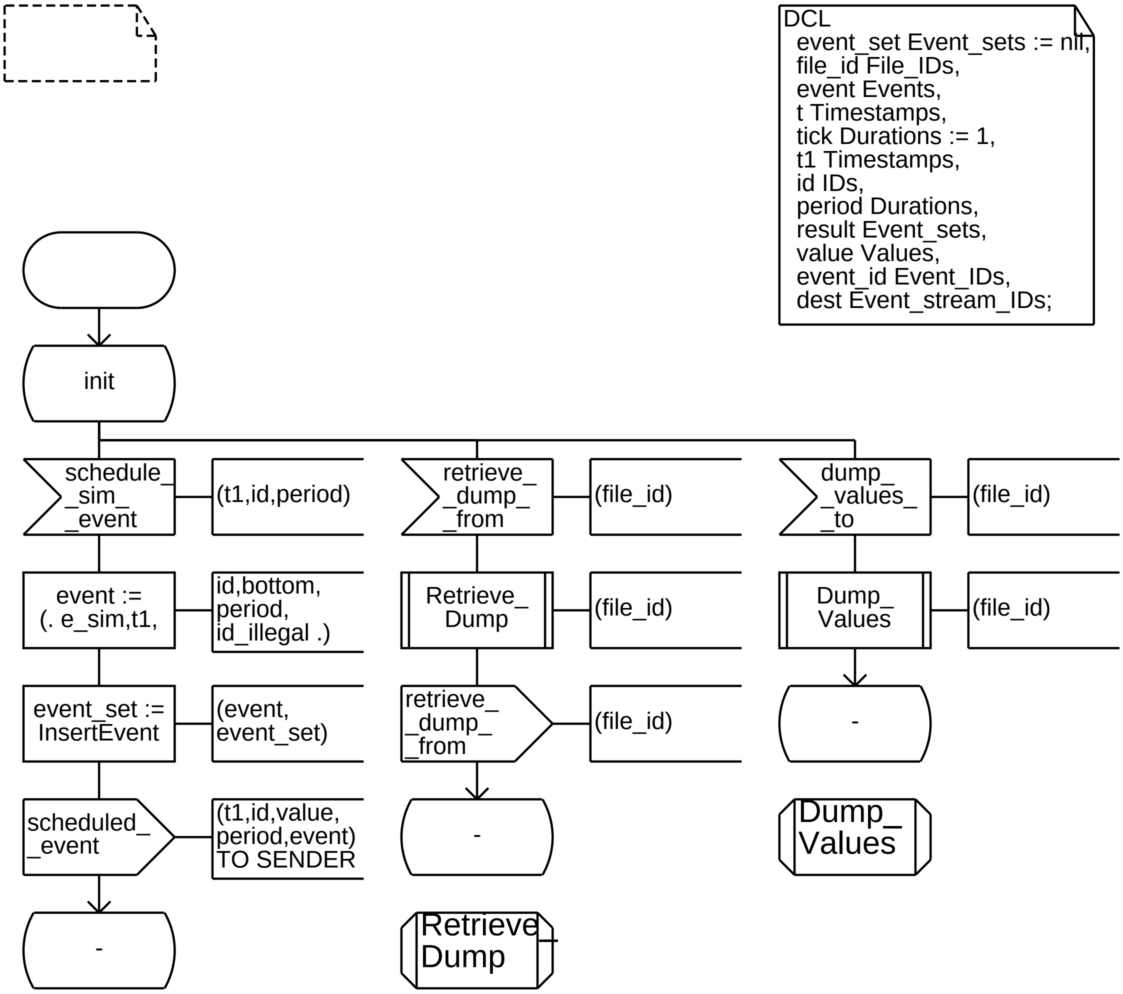
PROCEDURE <<PROCESS Clock>> RetrieveDump 1(2)



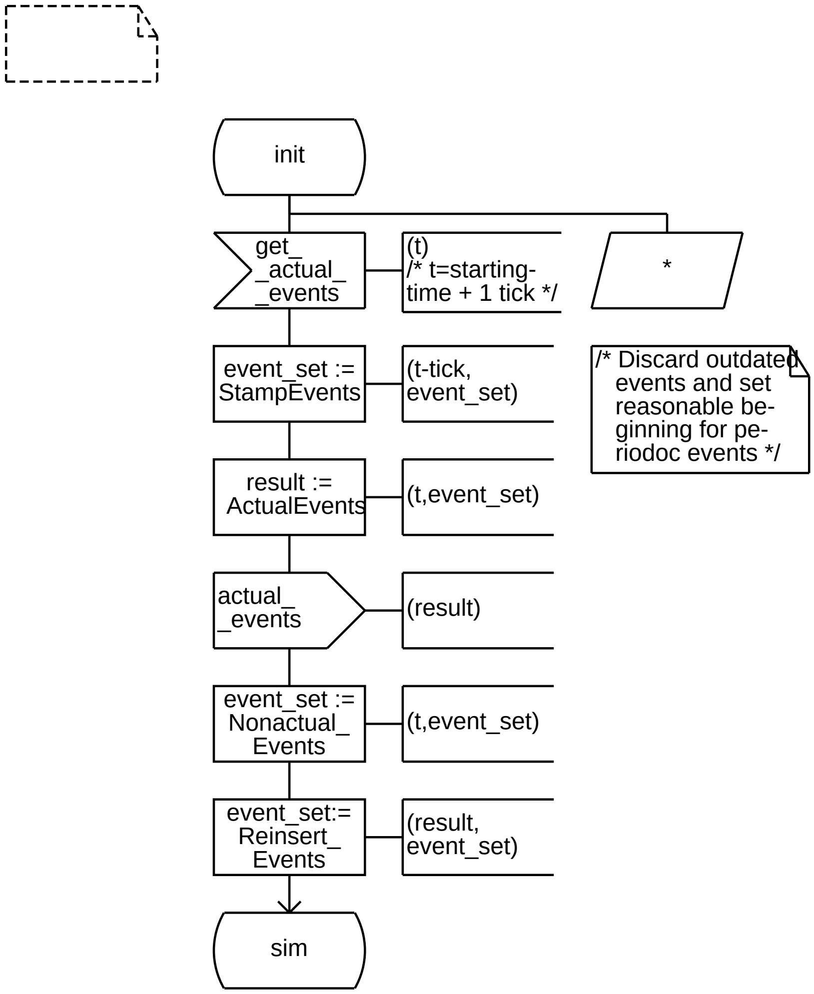
PROCEDURE <<PROCESS Clock>> RetrieveDump

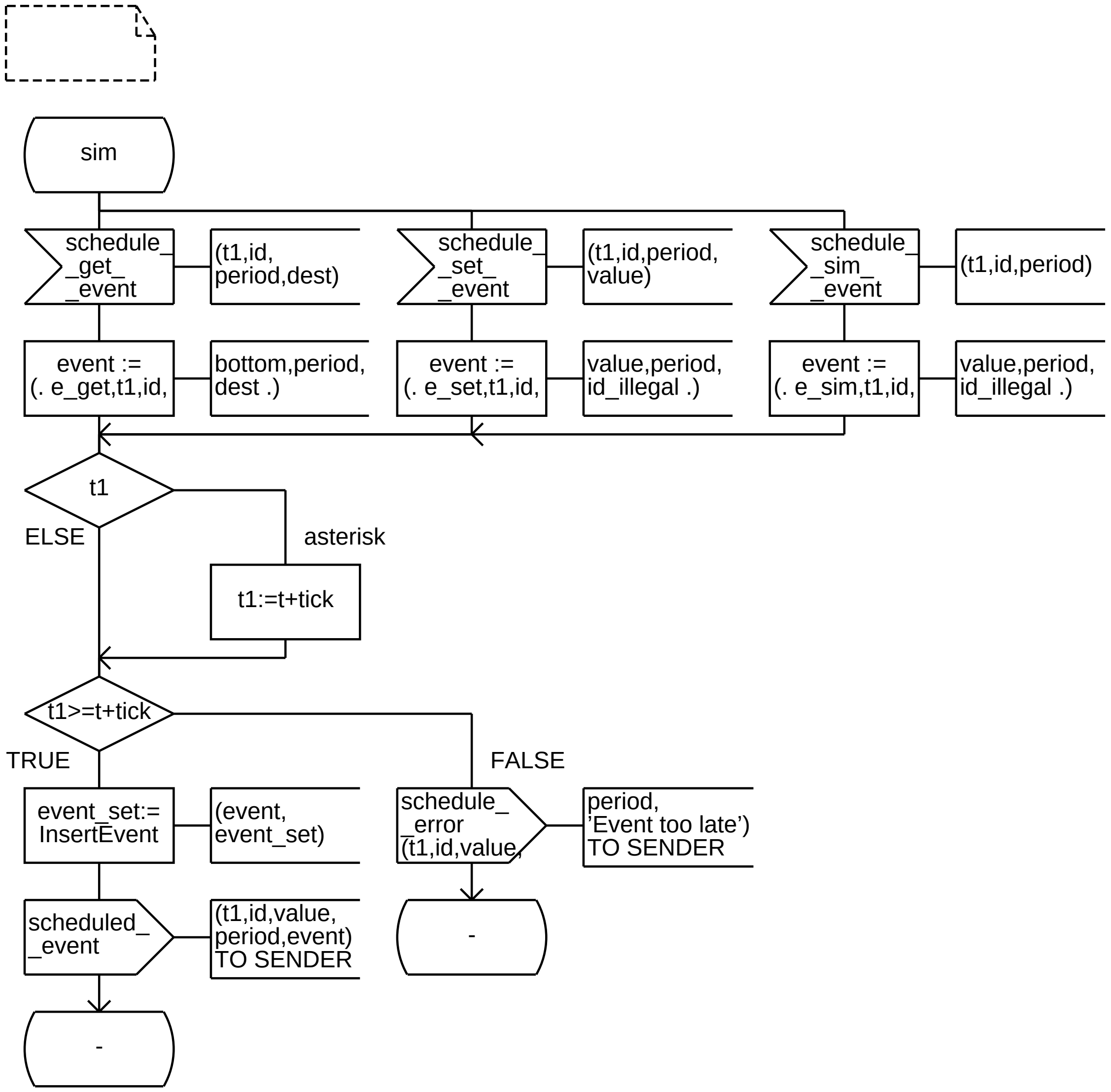


PROCESS EventCollection 1(4)

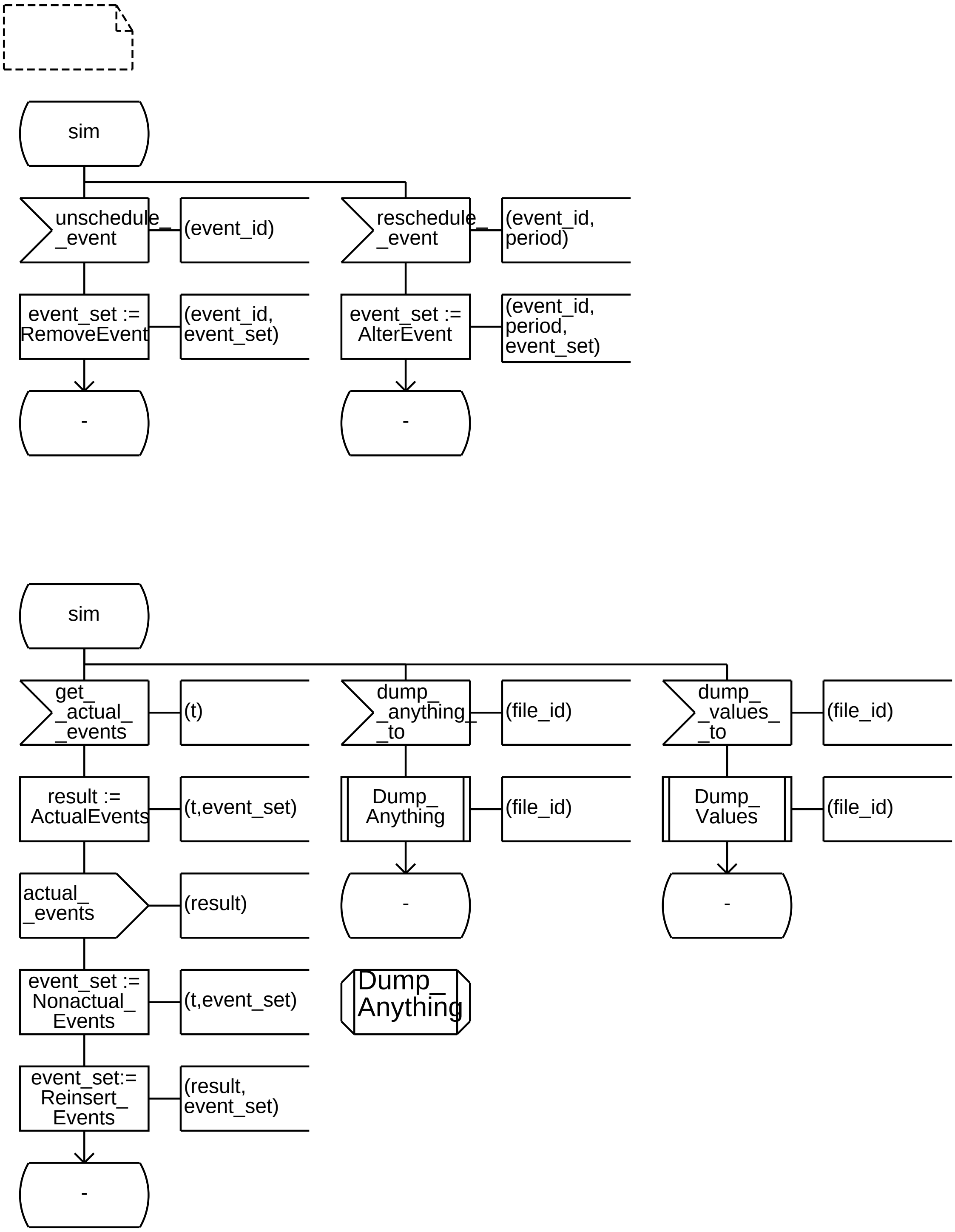


PROCESS EventCollection

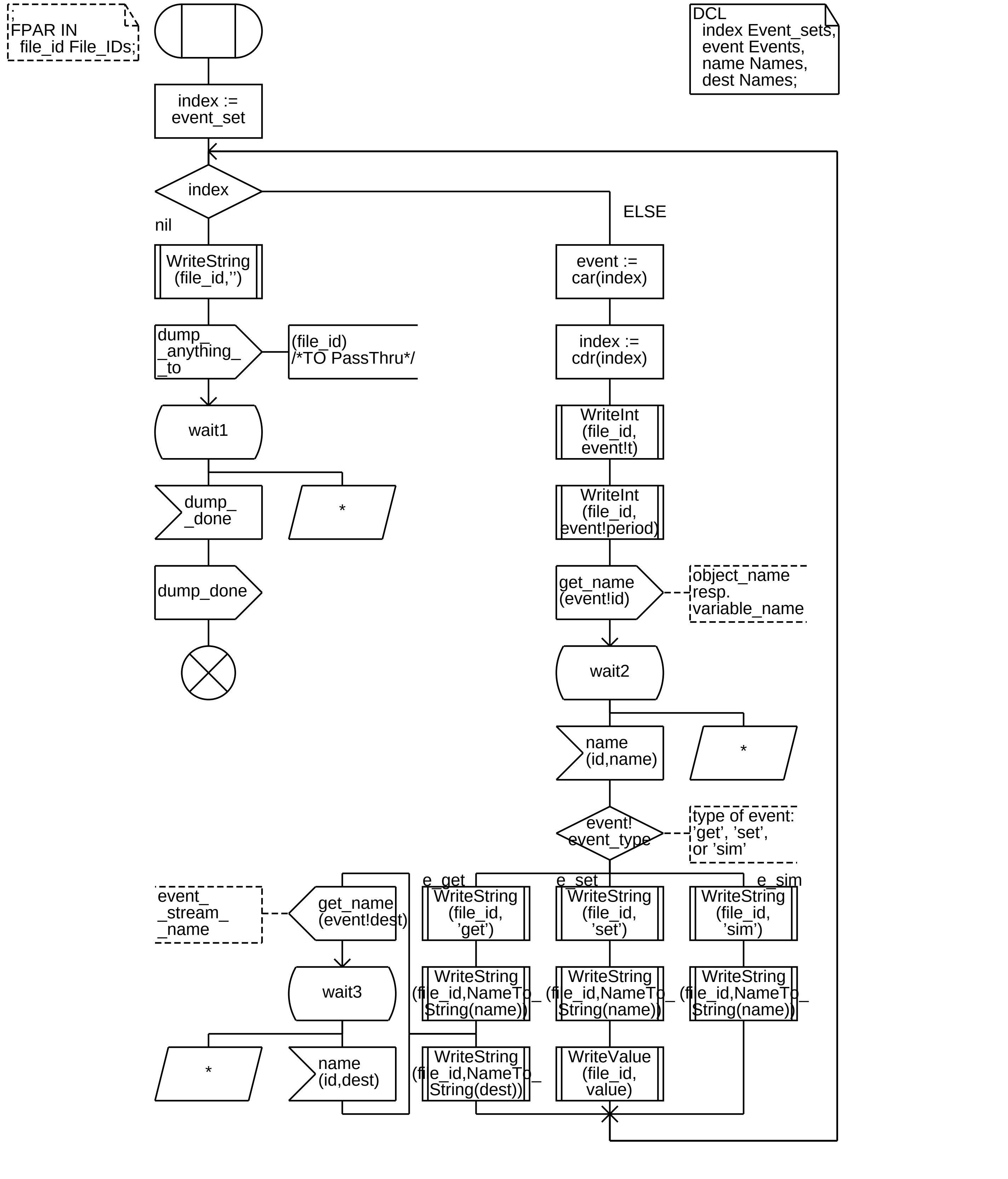



sim
schedule_
get
_event
(t1,id,
period,dest)
schedule_
set
_event
(t1,id,period,
value)
schedule_
sim
_event
(t1,id,period)
event :=
(. e_get,t1,id,
bottom,period,
dest .)
event :=
(. e_set,t1,id,
value,period,
id_illegal .)
event :=
(. e_sim,t1,id,
value,period,
id_illegal .)
t1
ELSE
asterisk
t1:=t+tick
t1>=t+tick
TRUE
FALSE
event_set:=
InsertEvent
(event,
event_set)
schedule_
_error
(t1,id,value,
period,
'Event too late')
TO SENDER
scheduled_
_event
(t1,id,value,
period,event)
TO SENDER
-
-

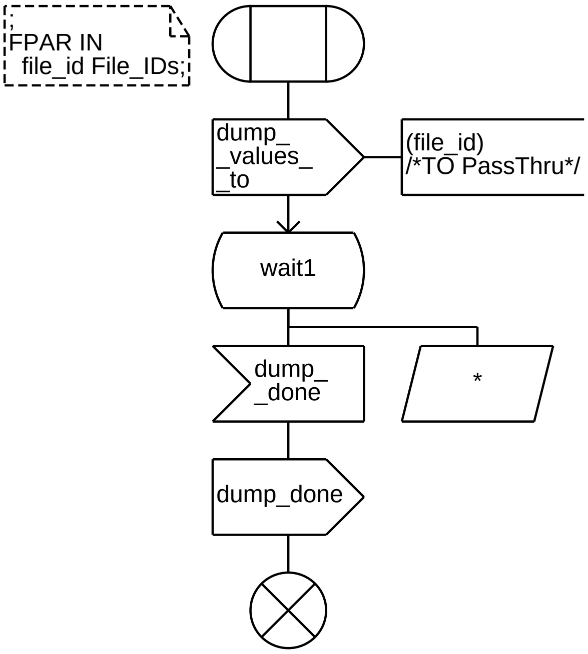
PROCESS EventCollection 4(4)

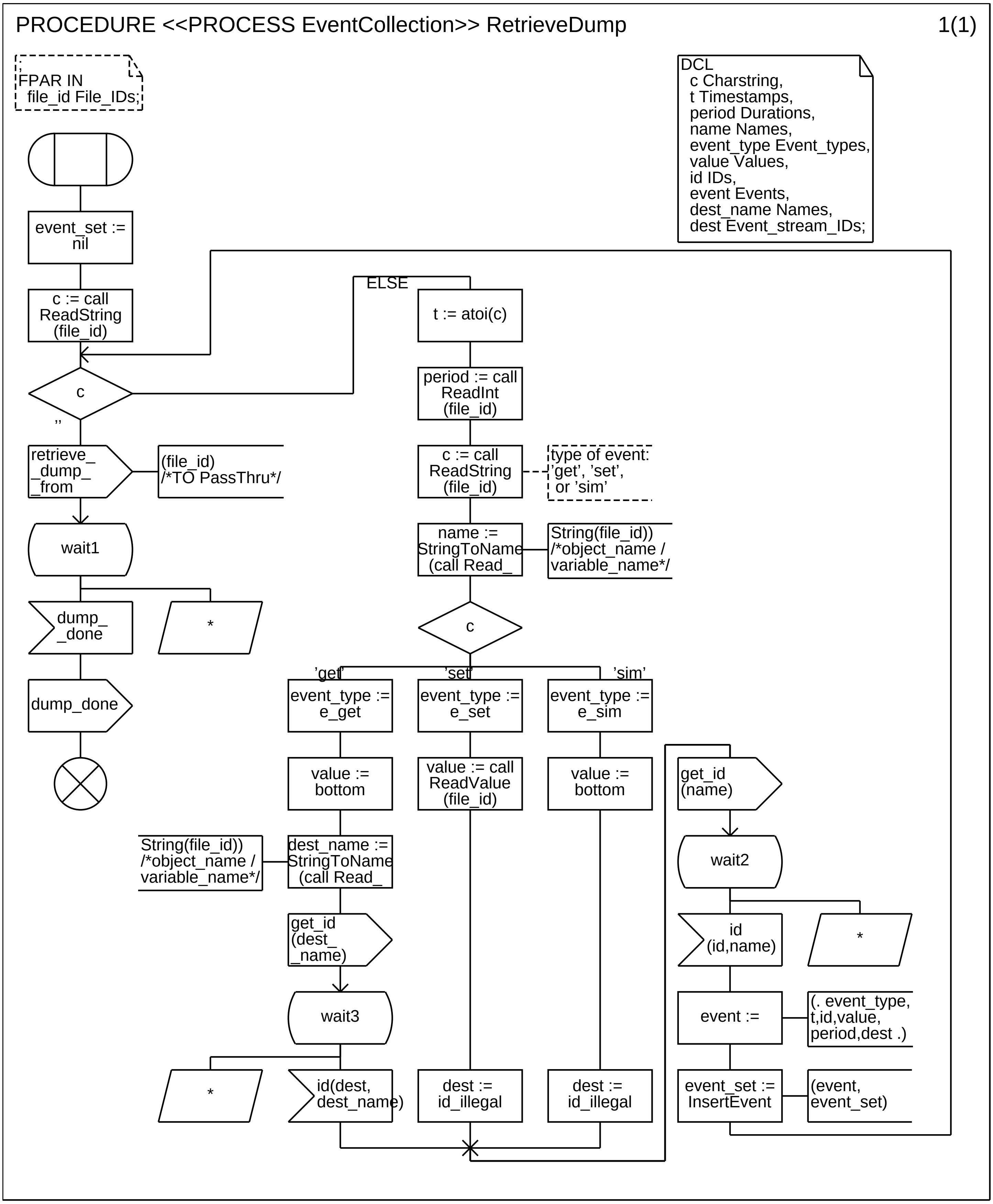


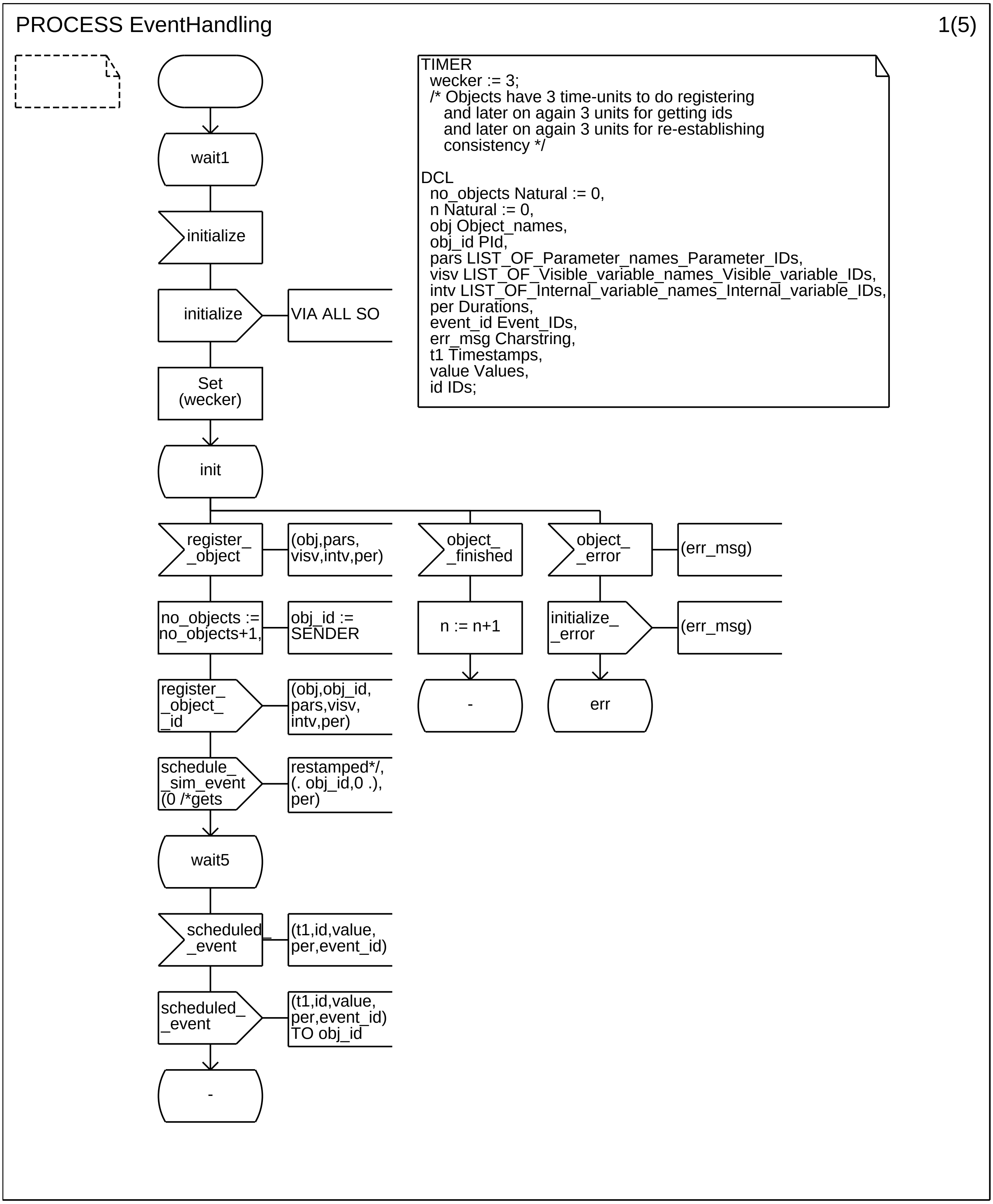
PROCEDURE <<PROCESS EventCollection>> DumpAnything 1(1)

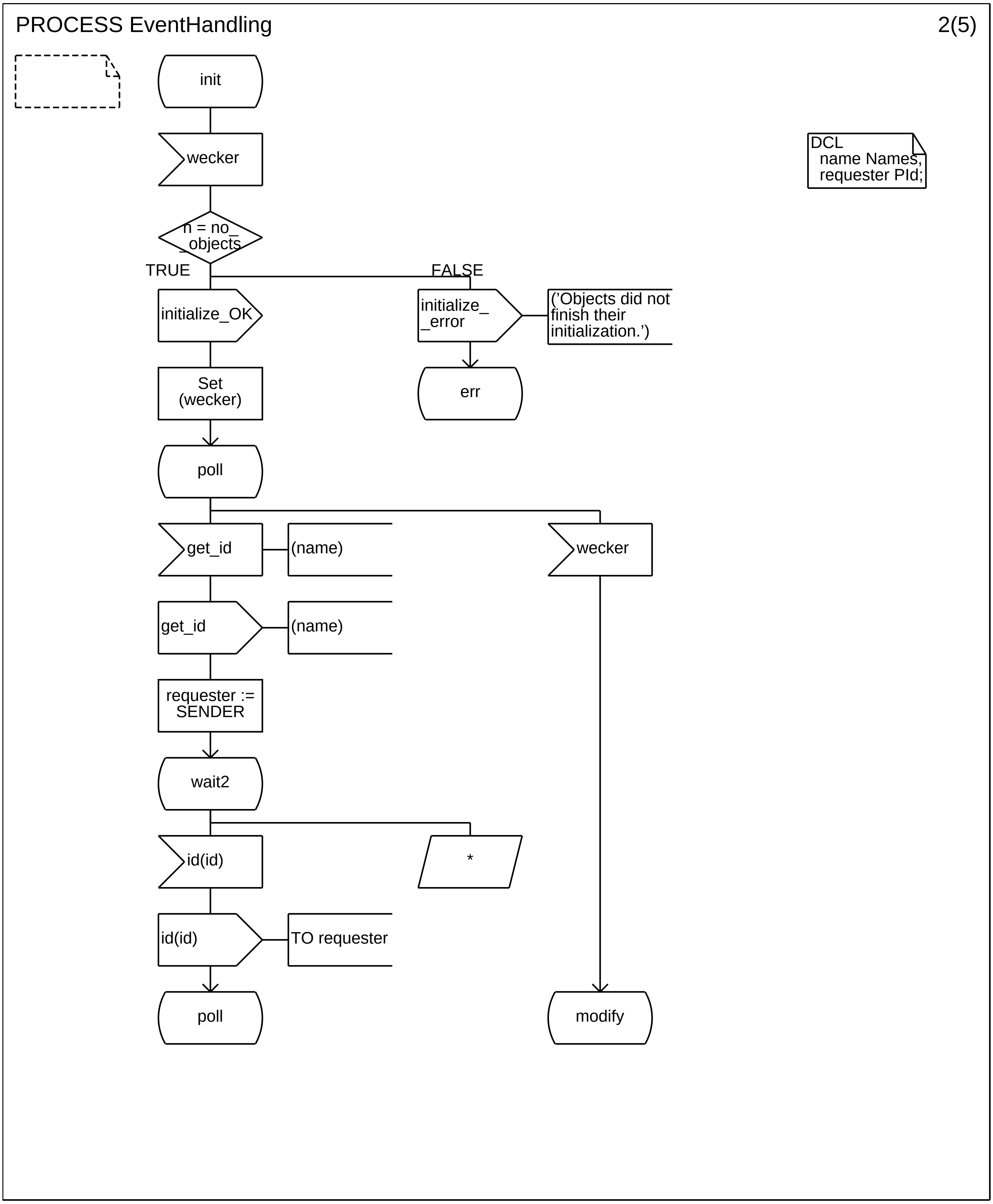


PROCEDURE <<PROCESS EventCollection>> DumpValues 1(1)



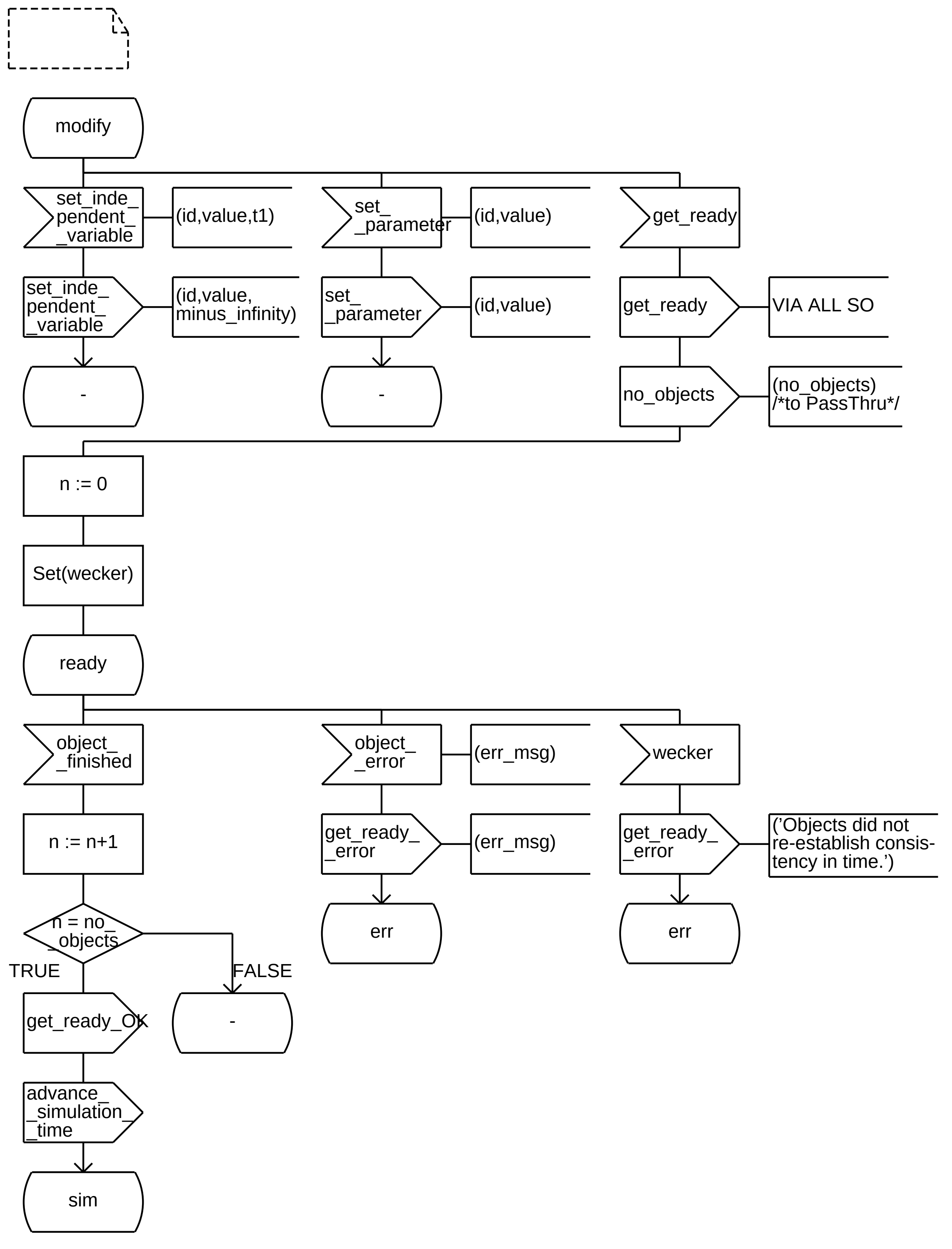

PROCEDURE <<PROCESS EventCollection>> RetrieveDump
1(1)
;
FPAR IN
file_id File_IDs;
DCL
c Charstring,
t Timestamps,
period Durations,
name Names,
event_type Event_types,
value Values,
id IDs,
event Events,
dest_name Names,
dest Event_stream_IDs;
event_set :=
nil
c := call
ReadString
(file_id)
c
''
ELSE
retrieve_
dump
_from
(file_id)
/*TO PassThru*/
wait1
dump_
_done
*
dump_done
t := atoi(c)
period := call
ReadInt
(file_id)
c := call
ReadString
(file_id)
type of event:
'get', 'set',
or 'sim'
name :=
StringToName
(call Read_
String(file_id))
/*object_name /
variable_name*/
c
'get'
'set'
'sim'
event_type :=
e_get
event_type :=
e_set
event_type :=
e_sim
value :=
bottom
value := call
ReadValue
(file_id)
value :=
bottom
get_id
(name)
String(file_id))
/*object_name /
variable_name*/
dest_name :=
StringToName
(call Read_
wait2
get_id
(dest_
_name)
id
(id,name)
*
wait3
event :=
(. event_type,
t,id,value,
period,dest .)
*
id(dest,
dest_name)
dest :=
id_illegal
dest :=
id_illegal
event_set :=
InsertEvent
(event,
event_set)

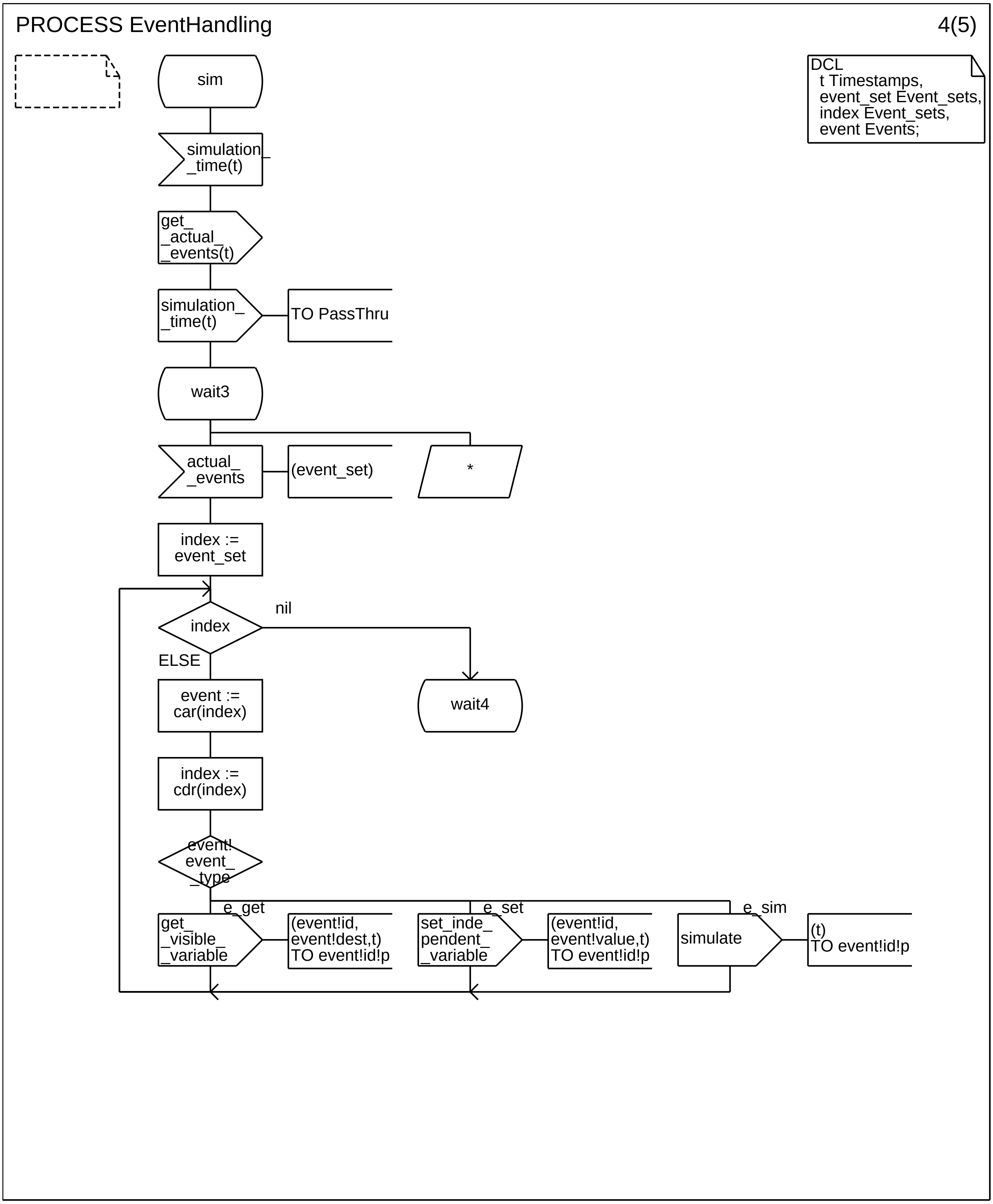

PROCESS EventHandling
1(5)
TIMER
wecker := 3;
/* Objects have 3 time-units to do registering
and later on again 3 units for getting ids
and later on again 3 units for re-establishing
consistency */
DCL
no_objects Natural := 0,
n Natural := 0,
obj Object_names,
obj_id PId,
pars LIST_OF_Parameter_names_Parameter_IDs,
visv LIST_OF_Visible_variable_names_Visible_variable_IDs,
intv LIST_OF_Internal_variable_names_Internal_variable_IDs,
per Durations,
event_id Event_IDs,
err_msg Charstring,
t1 Timestamps,
value Values,
id IDs;
wait1
initialize
initialize
VIA ALL SO
Set (wecker)
init
register_ _object
(obj,pars, visv,intv,per)
object_ _finished
object_ _error
(err_msg)
no_objects := no_objects+1,
obj_id := SENDER
n := n+1
initialize_ _error
(err_msg)
register_ _object_ _id
(obj,obj_id, pars,visv, intv,per)
-
err
schedule_ _sim_event (0 /*gets
restamped*/, (. obj_id,0 .), per)
wait5
scheduled_ _event
(t1,id,value, per,event_id)
scheduled_ _event
(t1,id,value, per,event_id) TO obj_id
-



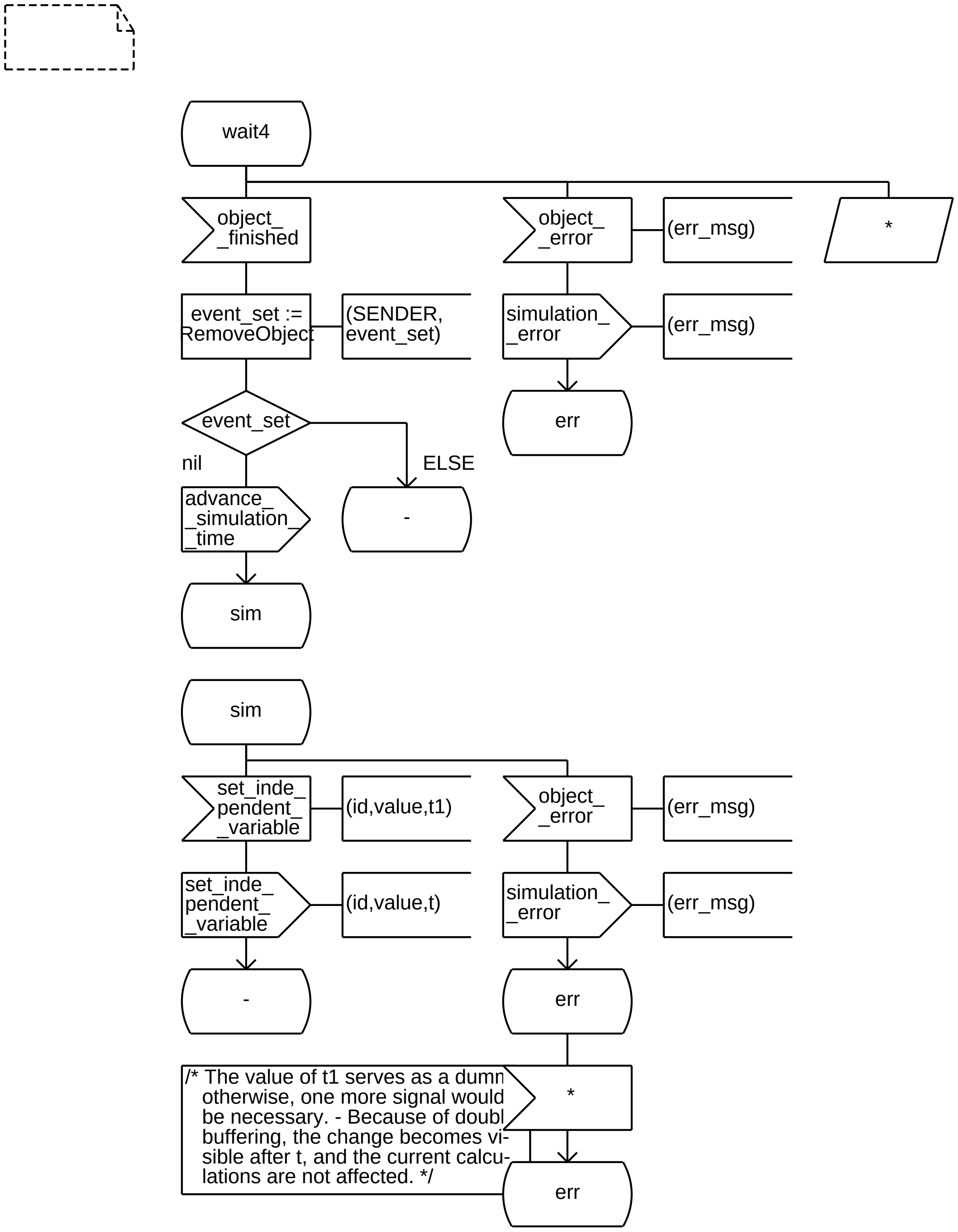
PROCESS EventHandling
2(5)
init
wecker
n = no_
_objects
TRUE
FALSE
initialize_OK
initialize_
_error
('Objects did not
finish their
initialization.')
err
Set
(wecker)
poll
get_id
(name)
wecker
get_id
(name)
requester :=
SENDER
wait2
id(id)
*
id(id)
TO requester
poll
modify
DCL
name Names;
requester PId;

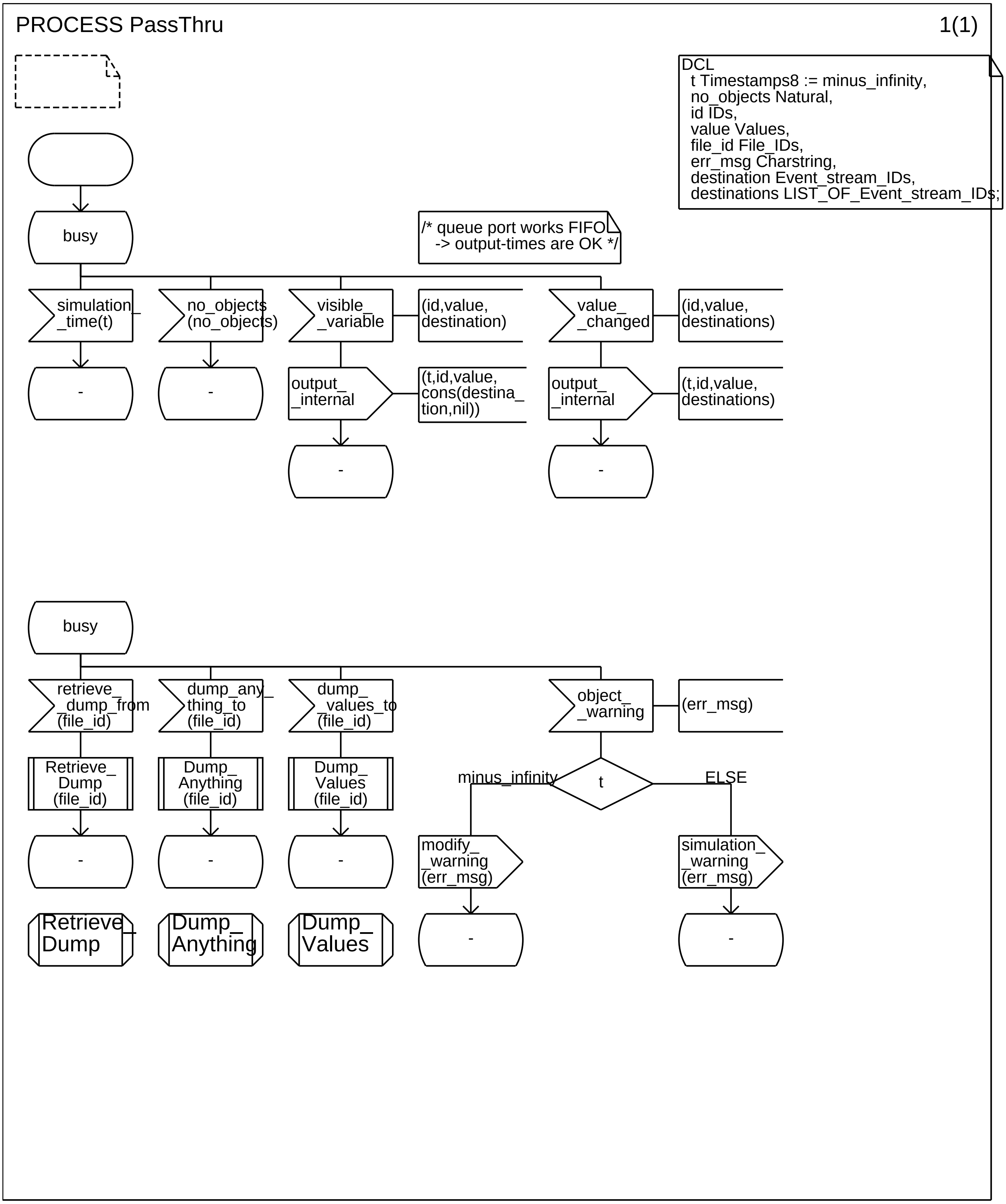
PROCESS EventHandling



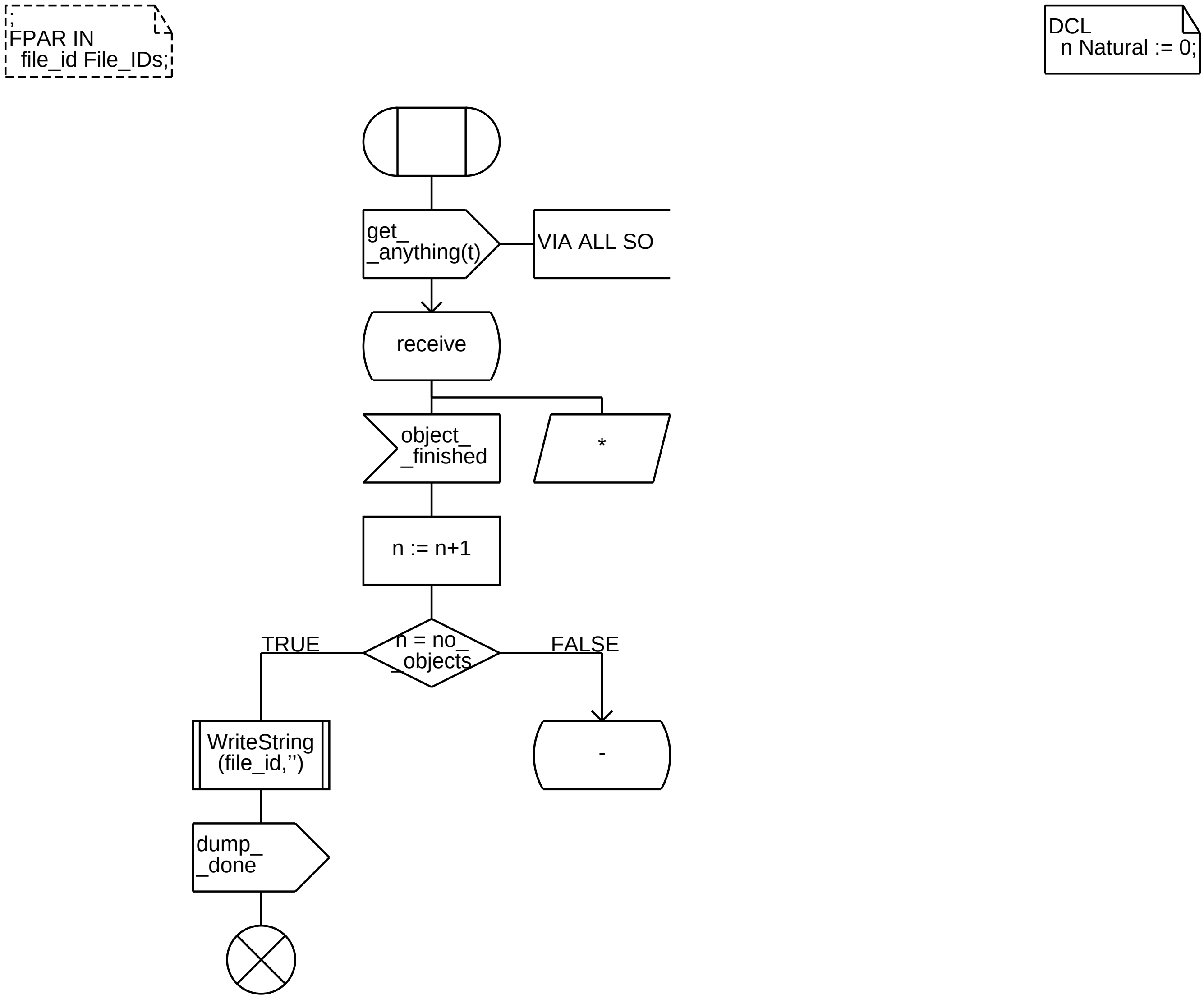

PROCESS EventHandling
4(5)
DCL
t Timestamps,
event_set Event_sets,
index Event_sets,
event Events;
sim
simulation_
_time(t)
get_
actual
_events(t)
simulation_
_time(t)
TO PassThru
wait3
actual_
_events
(event_set)
*
index :=
event_set
index
nil
ELSE
wait4
event :=
car(index)
index :=
cdr(index)
event!
event_
_type
e_get
e_set
e_sim
get_
visible
_variable
(event!id,
event!dest,t)
TO event!id!p
set_inde_
pendent_
_variable
(event!id,
event!value,t)
TO event!id!p
simulate
(t)
TO event!id!p

PROCESS EventHandling

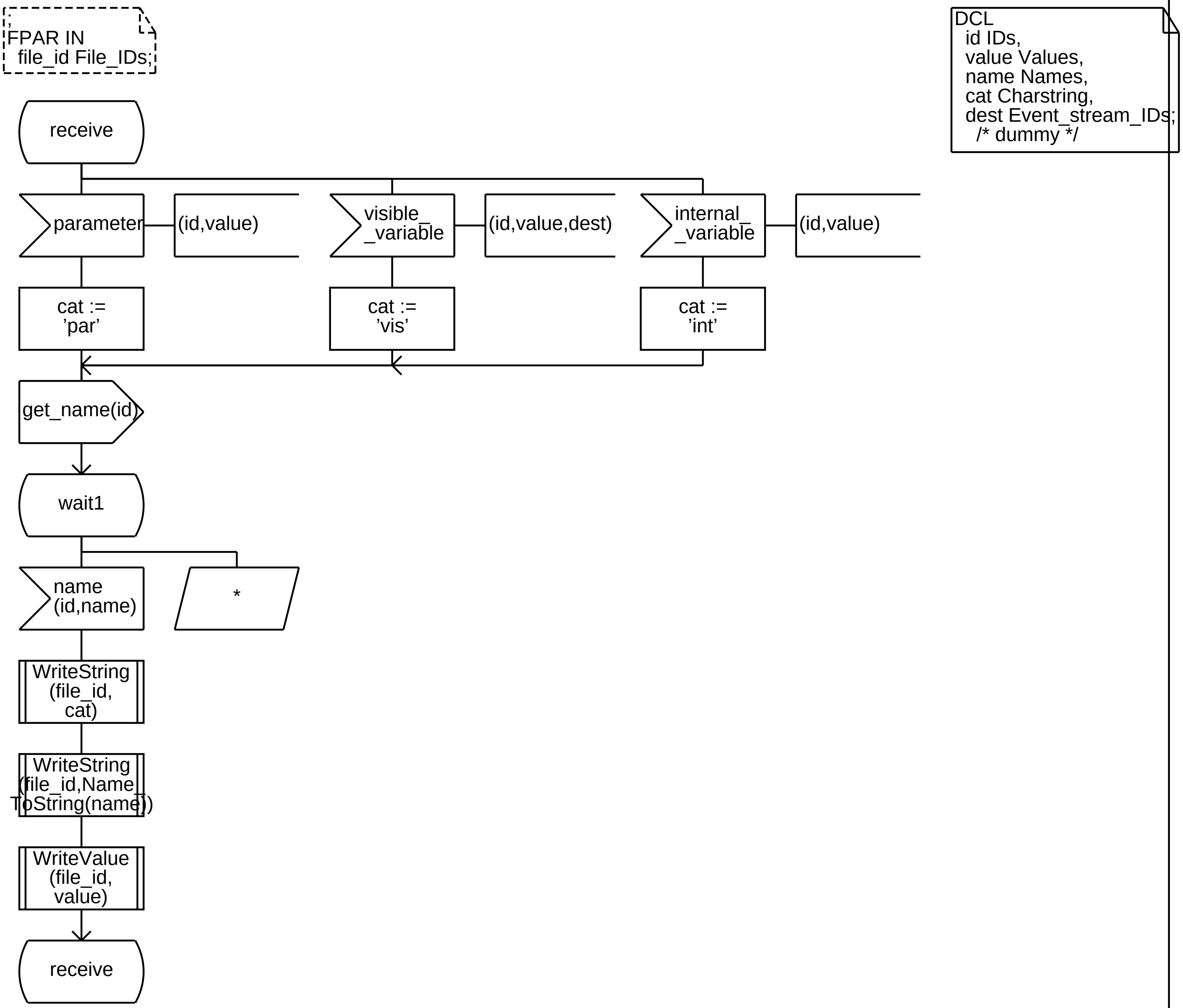


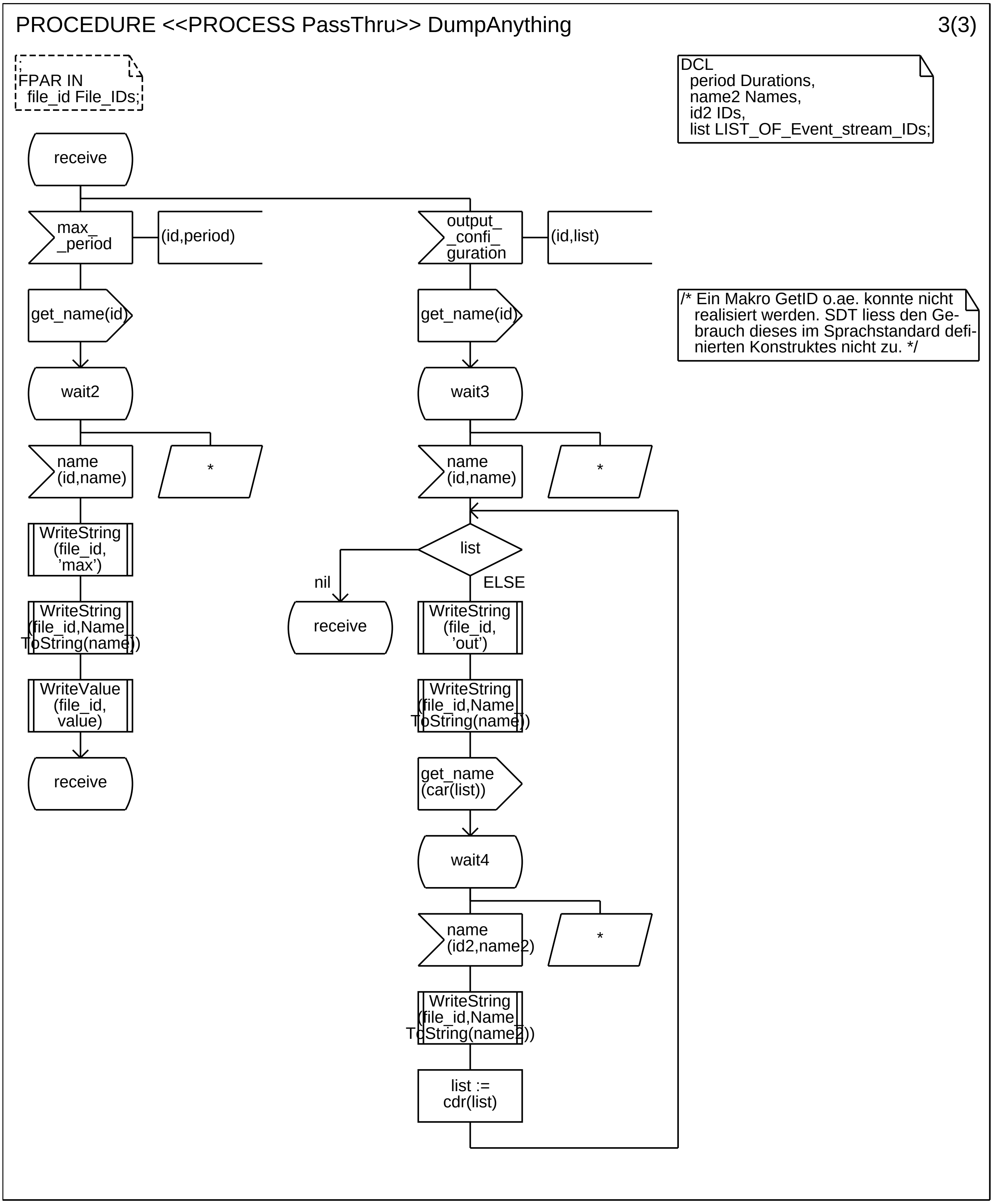

PROCESS PassThru
1(1)
DCL
t Timestamps8 := minus_infinity,
no_objects Natural,
id IDs,
value Values,
file_id File_IDs,
err_msg Charstring,
destination Event_stream_IDs,
destinations LIST_OF_Event_stream_IDs;
busy
/* queue port works FIFO
-> output-times are OK */
simulation_
_time(t)
no_objects
(no_objects)
visible_
_variable
(id,value,
destination)
value_
_changed
(id,value,
destinations)
-
-
output_
_internal
(t,id,value,
cons(destina_
tion,nil))
output_
_internal
(t,id,value,
destinations)
-
-
busy
retrieve_
_dump_from
(file_id)
dump_any_
thing_to
(file_id)
dump_
_values_to
(file_id)
object_
_warning
(err_msg)
Retrieve_
Dump
(file_id)
Dump_
Anything
(file_id)
Dump_
Values
(file_id)
minus_infinity
t
ELSE
-
-
-
modify_
_warning
(err_msg)
simulation_
_warning
(err_msg)
Retrieve_
Dump
Dump_
Anything
Dump_
Values
-
-

PROCEDURE <<PROCESS PassThru>> DumpAnything 1(3)

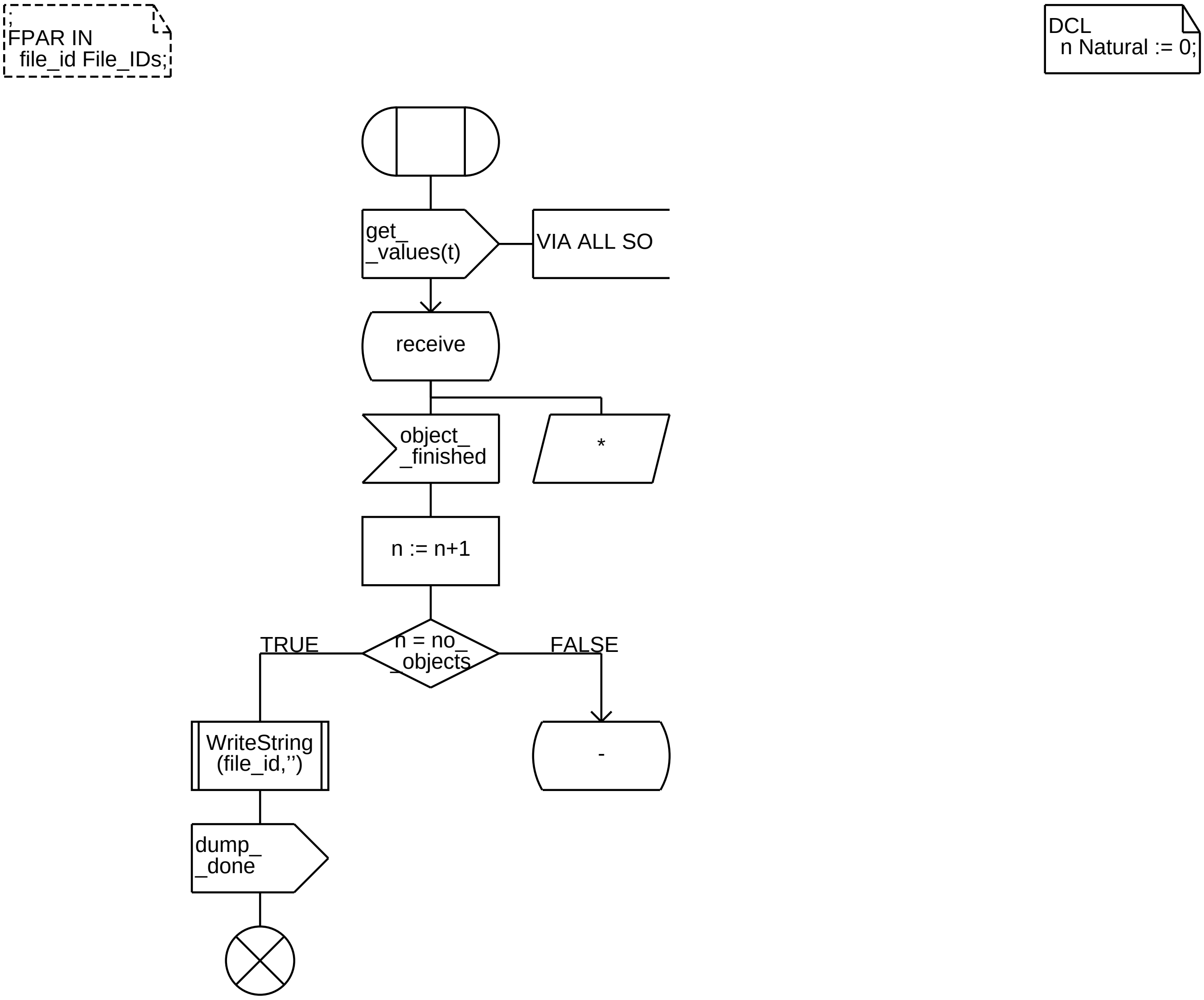


PROCEDURE <<PROCESS PassThru>> DumpAnything 2(3)

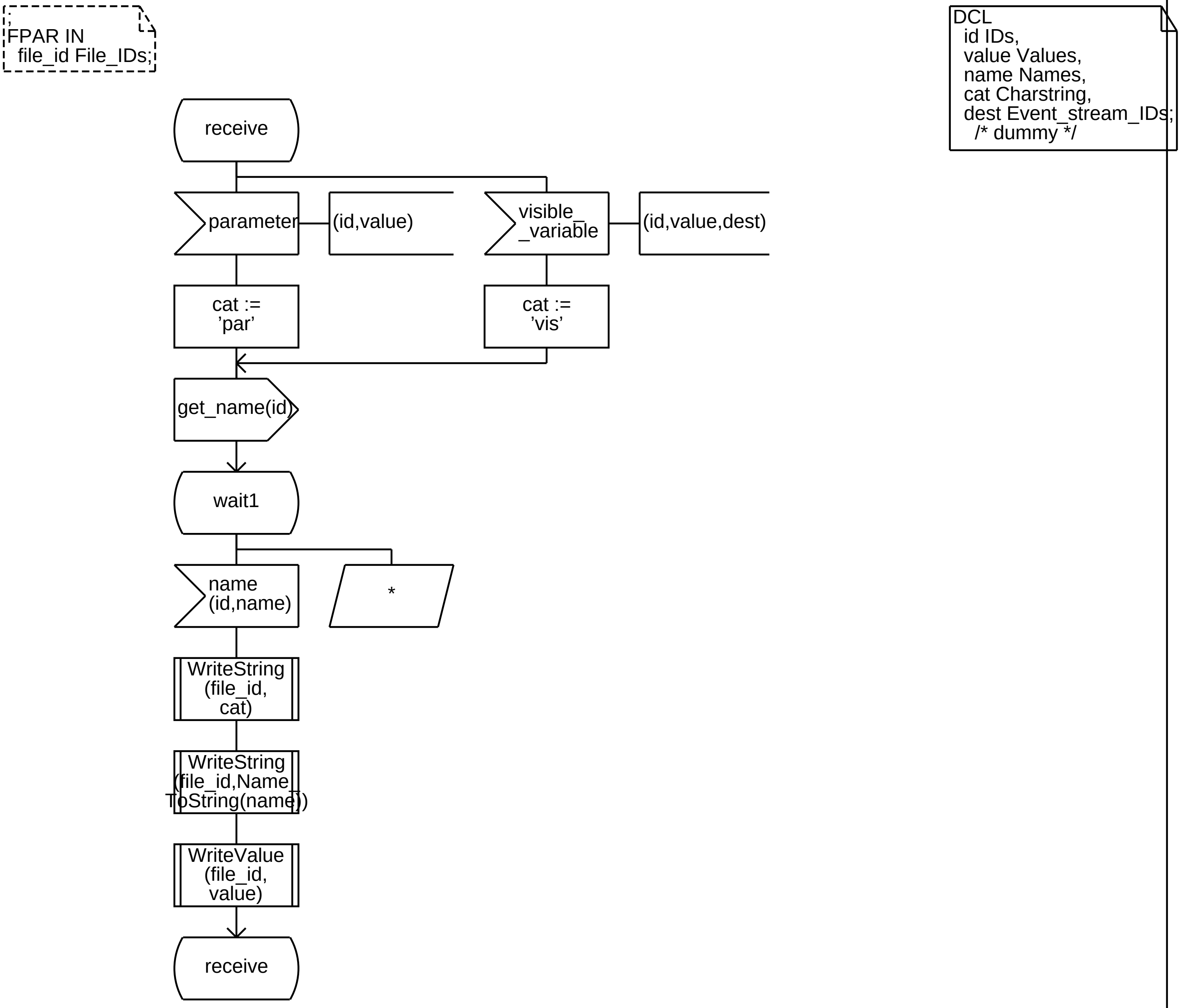


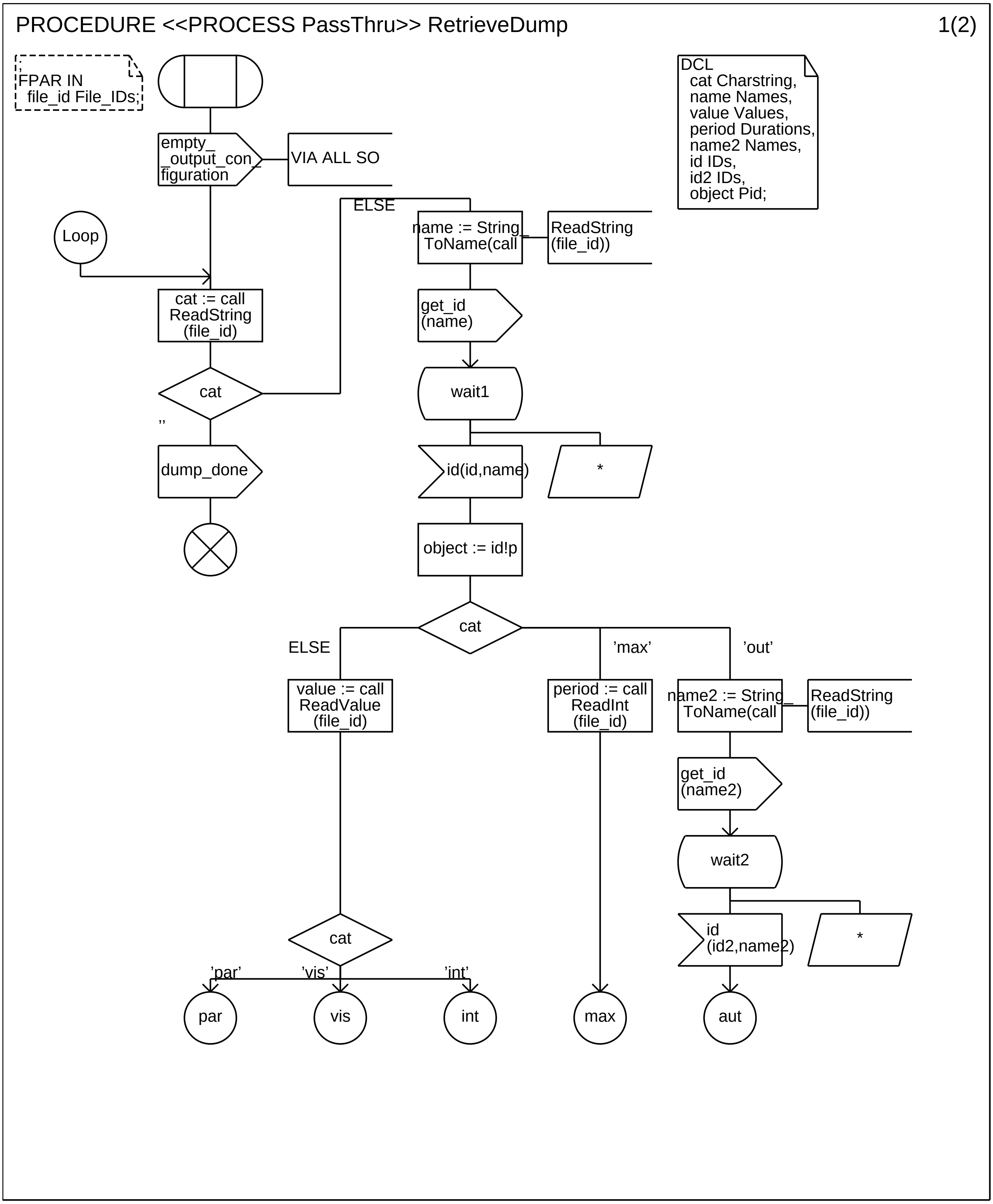

PROCEDURE <<PROCESS PassThru>> DumpAnything
3(3)
;
FPAR IN
file_id File_IDs;
DCL
period Durations,
name2 Names,
id2 IDs,
list LIST_OF_Event_stream_IDs;
/* Ein Makro GetID o.ae. konnte nicht realisiert werden. SDT liess den Gebrauch dieses im Sprachstandard definierten Konstruktes nicht zu. */
receive
max_ _period
(id,period)
get_name(id)
wait2
name (id,name)
*
WriteString (file_id, 'max')
WriteString (file_id,Name ToString(name))
WriteValue (file_id, value)
receive
output_ _confi_ guration
(id,list)
get_name(id)
wait3
name (id,name)
*
list
nil
ELSE
receive
WriteString (file_id, 'out')
WriteString (file_id,Name ToString(name))
get_name (car(list))
wait4
name (id2,name2)
*
WriteString (file_id,Name ToString(name2))
list := cdr(list)

PROCEDURE <<PROCESS PassThru>> DumpValues 1(2)



PROCEDURE <<PROCESS PassThru>> DumpValues




PROCEDURE <<PROCESS PassThru>> RetrieveDump
1(2)
FPAR IN
file_id File_IDs;
DCL
cat Charstring,
name Names,
value Values,
period Durations,
name2 Names,
id IDs,
id2 IDs,
object Pid;
empty_
_output_con_
figuration
VIA ALL SO
Loop
cat := call
ReadString
(file_id)
cat
''
dump_done
ELSE
name := String_
ToName(call
ReadString
(file_id))
get_id
(name)
wait1
id(id,name)
*
object := id!p
cat
ELSE
'max'
'out'
value := call
ReadValue
(file_id)
period := call
ReadInt
(file_id)
name2 := String_
ToName(call
ReadString
(file_id))
get_id
(name2)
wait2
id
(id2,name2)
*
cat
'par'
'vis'
'int'
par
vis
int
max
aut

PROCEDURE <<PROCESS PassThru>> RetrieveDump