REAL TIME SYSTEMS

3. APEX : ARINC 653 standard

Introduction

Specify an interface between an OS of an avionics resource and application software: APEX interface (APplication / EXecutive)

Start of work: 1991

Publication of the first ARINC 653 standard: Summer 1996

Definitions

Application :

— Software realization of an avionics function

— consisting of one or more partitions

Example: Automatic Pilot

Partition :

- Entity of execution of an application
- Runs on a single processor
- Deterministic allocation of resources to partitions
- Spatial (memory) and temporal (CPU) segregation unit
- Consisting of one or more processes

Example : FM1, FM2, FE1-COM, FE1-MON, FE2-COM, FE2-MON, FG1-COM, FG1-MON, FG2-COM, FG2-MON

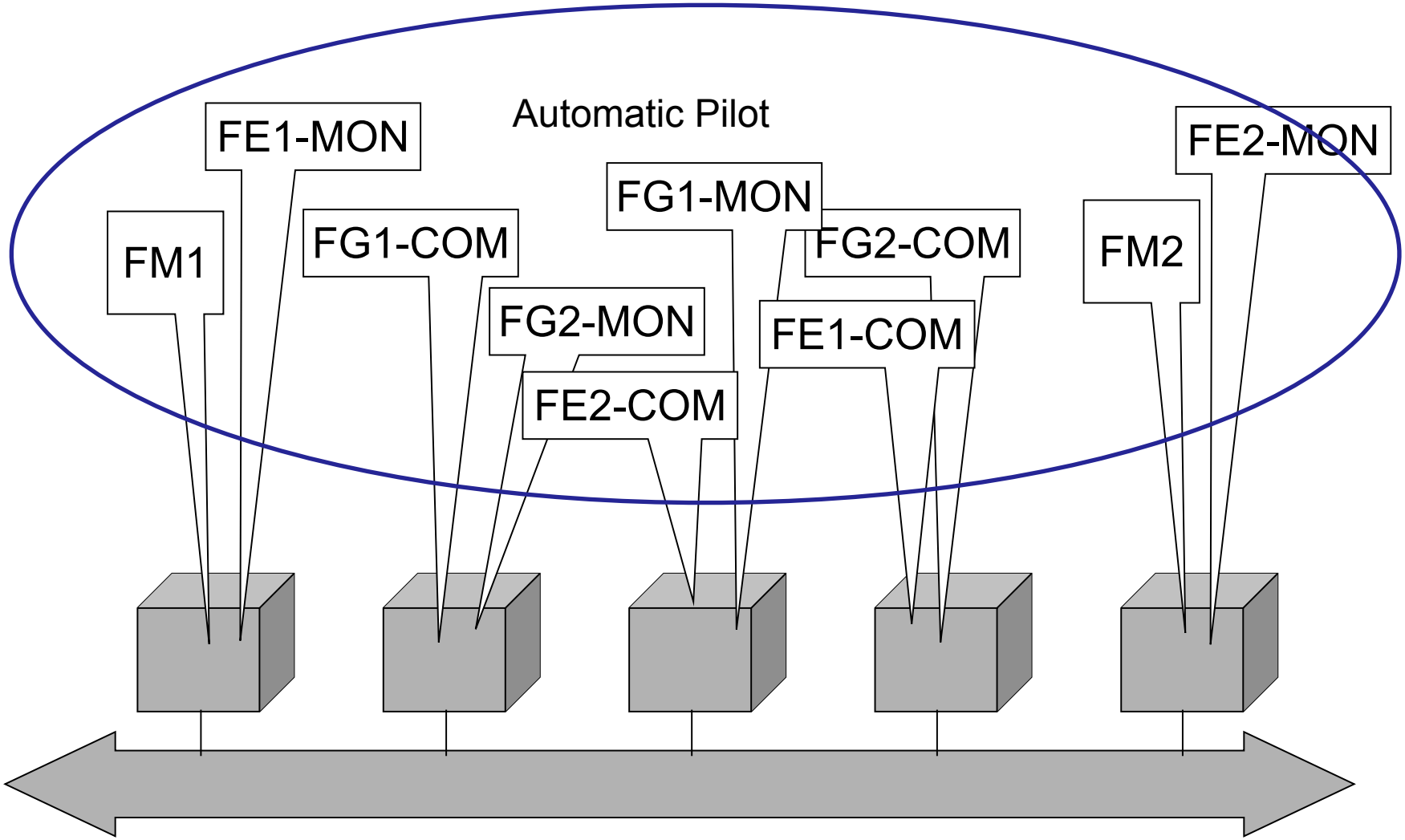
Definition

Process:

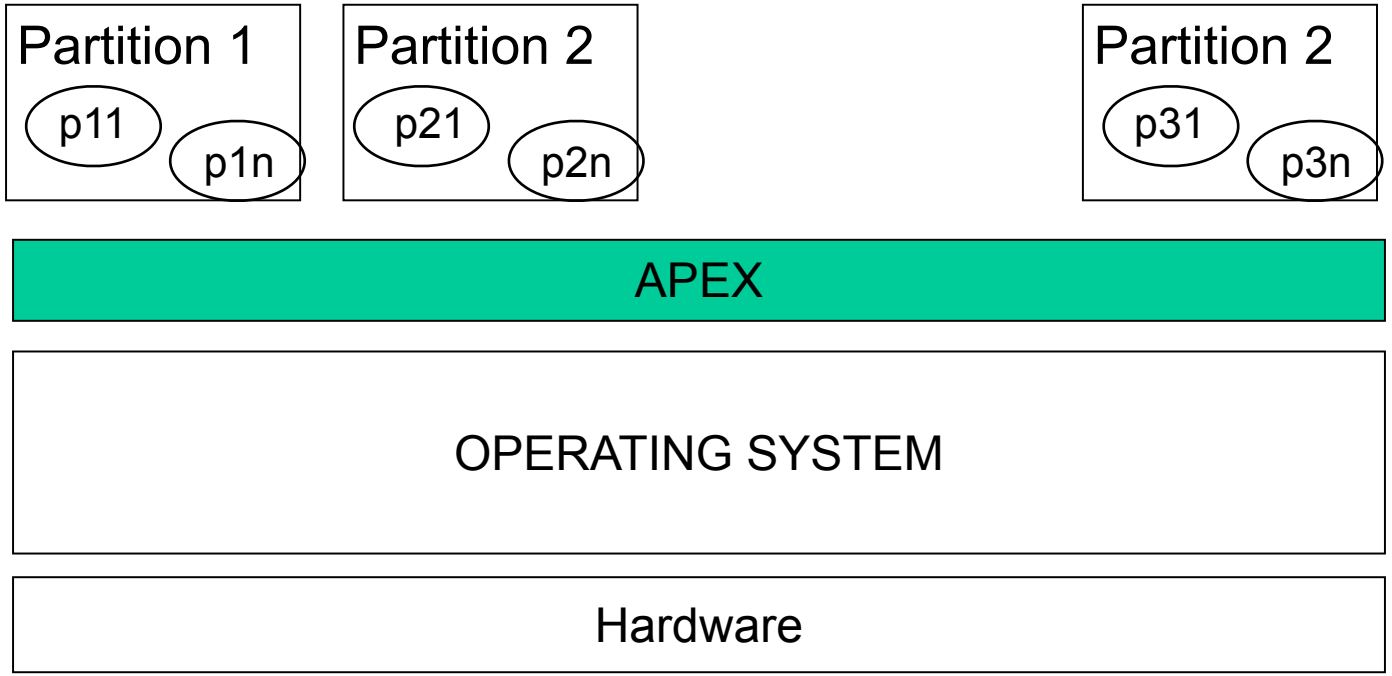
- A program unit that runs in the partition environment.
- Concurrent execution of processes to perform the avionics function

Example: a control law, a control logic…

Architecture



Architecture



Operating system :

- Schedules the partitions of the module
- Schedules the processes in the partition
- Ensures spatial and temporal segregation (partitioning)

Resource Sharing

Spatial sharing:

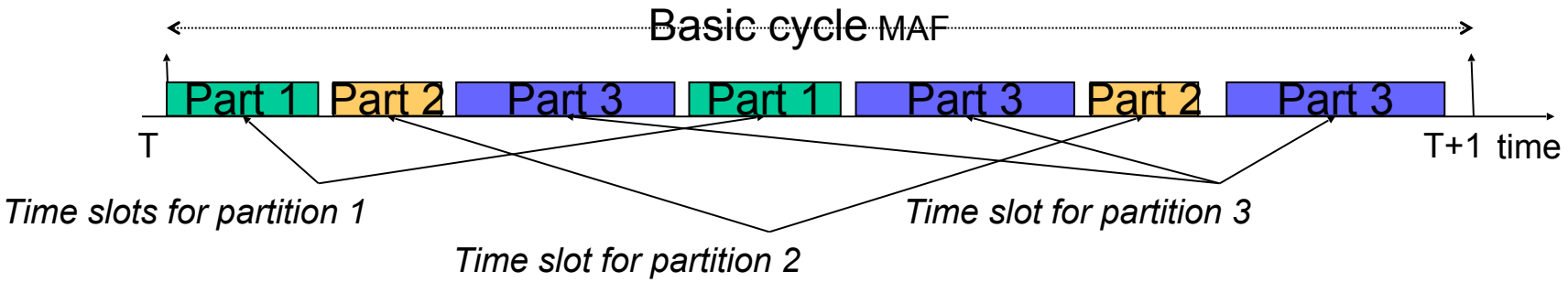
Predetermined memory area for each partition

Time sharing (CPU):

- A partition has no priority
- Deterministic and cyclic allocation of the processor to partitions:

  => The OS repeats a basic cycle (MAjor time Frame: MAF) of fixed duration.

  => Assign one or more time slots in the MAF to each partition.



=> Resource allocation is defined by configuration and cannot be changed dynamically.

Partitions management

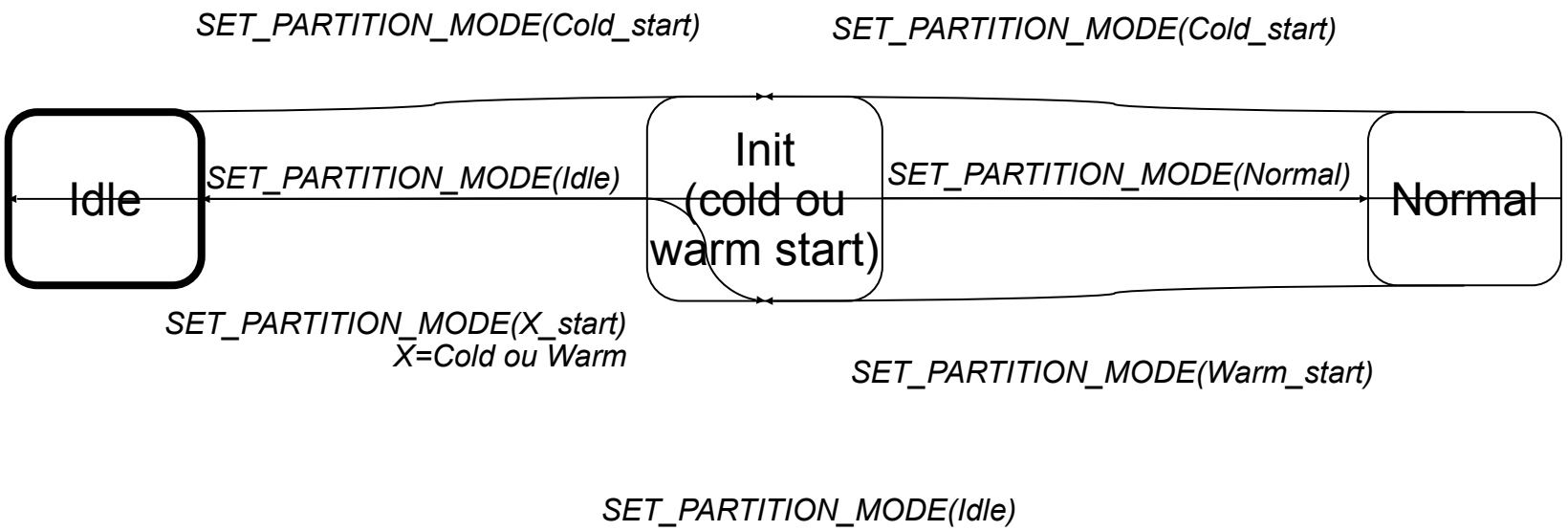
- The OS starts the partition scheduler at the end of module initialization.
- During its time slots, a partition has the following states
  - Idle :
    - No process is running.
  - Cold_start ou Warm_start : initialization of the partition
    - Cold start: abstraction of the previous context of the partition
    - Warm start: recovering the previous context of the partition
  - Normal : starting process scheduling
- Outside its time slots a partition is suspended.
- The partition creates all its objects (ports, process, semaphores...) during its initialization phase.
- The partition starts process scheduling at the end of its initialization phase.

Partitions management

APEX service for partitions

SET_PARTITION_MODE: changes the mode (Idle, Cold or Warm start, Normal)

GET_PARTITION_STATUS: provides the status of the partition



Process management

- 2 process types
  - Periodic: execution in regular intervals
  - Apériodic : execution on event occurrence
- No segregation between processes within the same application
- Processes on one partition are not visible to other processes on other partitions.
- Each process has a priority
- Process scheduling is based on a preemption algorithm by priority level and current process state
- Only one process per partition is "running" at any given time.

Process management

APEX services for processes

CREATE_PROCESS :

- Creation of a link between the name of the process and its reservation on partition initialization
- Putting the process in a S*leeping* state and returning an identifier (id)

START :

- Initialization of the process
- Change from *Sleeping* to *Ready* state

STOP :

- stopping a process by removing access rights to the processor
- Change from *Ready* or *Waiting* to *Sleeping* state

STOP_SELF :

- Auto-stop of the running process
- Change from *Running* to *Sleeping* state

SUSPEND :

- Suspension of the process until resumed by another process
- Change from *Running* to *Waiting* state

SUSPEND_SELF :

- self-suspension of the process
- Change from *Running* to *Waiting* state

Process management

APEX service for processes

RESUME :

Restarting a suspended process

If the process is already waiting for a resource, it remains in the *Waiting* state, otherwise it switches to the *Ready* state

SET_PRIORITY :

Changing the priority of a process

LOCK_PREEMPTION :

Incrementing the preemption counter of the partition and blocking the preemption mechanism

UNLOCK_PREEMPTION :

Decrementing the preemption counter of the partition and unlocking the preemption mechanism if this counter is null

GET_PROCESS_ID :

returns the identifier of a process by giving its name

GET_PROCESS_STATUS :

returns status information about a process

Process management

Time management

Time is expressed in real time (absolute).

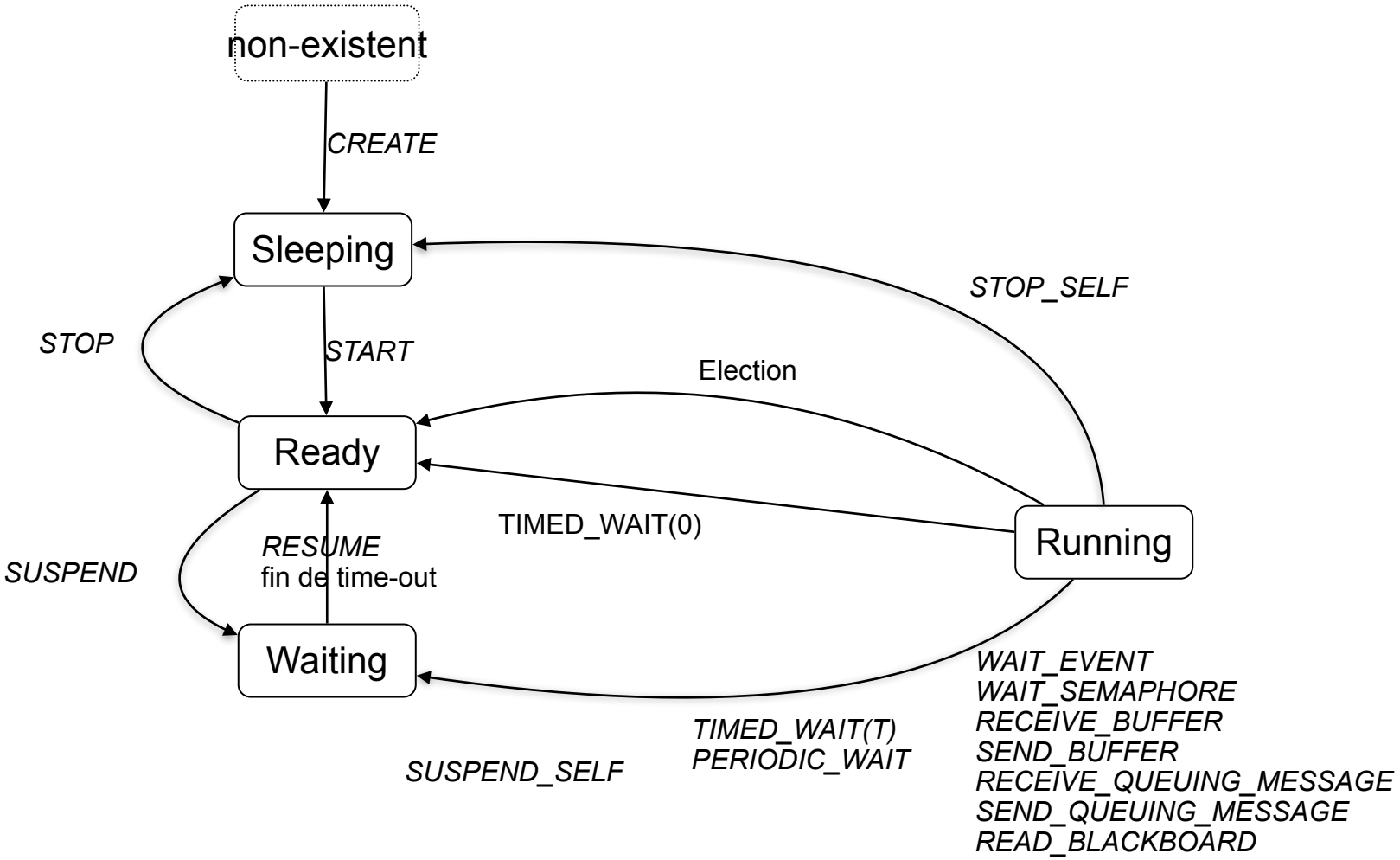
The OS ensures time segregation between partitions

APEX services for time management :

- TIMED_WAIT :
  - Suspension of a process for a given minimum amount of time
  - Change from Running to Waiting state
  - A time equal to zero performs a "Round Robin" on processes with the same priority.
- PERIODIC_WAIT
  - Suspension of a periodic process until the next period
  - reallocation of the process time budget at the beginning of the period
- GET_TIME
  - returns time (absolute) and local time to the module

Process management

State graph of the processes



Inter-process communication

Processes on the same partition can communicate and synchronize through

- communication via Buffer or Blackboard type mailboxes
- use of global variables to the partition
- synchronization by semaphore and event mechanisms

A mailbox allows the exchange of messages between several processes without indicating the names of the sending and receiving processes.

A memory area is reserved for the initialization of the partition for these communication / synchronization objects.

Inter-process communication

The "Buffer" type mailboxes:

communication by message that can carry different data
a message occurrence does not overwrite previous occurrences
messages are stored in FIFO message queues
the message tails are limited in length
a message queue can be full
processes trying to read from an empty message queue are put on hold
processes seeking to transmit in a full message queue are put on hold

Corresponding APEX services:

CREATE_BUFFER
SEND_BUFFER
RECEIVE_BUFFER
GET_BUFFER_ID
GET_BUFFER_STATUS } → a timeout can be specified

Inter-process communication

"Blackboard" type mailboxes:

- data write communication
- a data entry overwrites the previous data
- a written data remains displayed until the next data is written or a delete request is made.
- all processes waiting on an empty "Blackboard" are woken up when writing data (change from *Waiting* to *Ready* status)

Corresponding APEX services:

- CREATE_BLACKBOARD
- DISPLAY_ BLACKBOARD
- READ_ BLACKBOARD
- CLEAR_ BLACKBOARD ⟶ a timeout can be specified
- GET_ BLACKBOARD _ID
- GET_ BLACKBOARD _STATUS

Inter-process communication

Smaphores:

=> semaphores with counter

resource and signals the semaphore when it releases the resource

if counter > 0: the value represents the number of processes that can access the resource.

if counter = 0: resource unavailable

processes waiting on a semaphore are ordered either by FIFO or by priority

Corresponding APEX services:

- CREATE_SEMAPHORE
- WAIT_ SEMAPHORE ——→ a timeout can be specified
- SIGNAL_ SEMAPHORE
- GET_ SEMAPHORE _ID
- GET_ SEMAPHORE _STATUS

Inter-process communication

Events

=> event at level (UP and DOWN)

allows to synchronize processes on the partition

when an event is set to UP, all processes pending on that event are set to *Ready* and a new process scheduling takes place

when a process is waiting for an event set to DOWN, it goes to the *Waiting* state with possibly waiting for a time out.

Services APEX correspondants :

- CREATE_ EVENT
- SET_ EVENT
- RESET_ EVENT
- WAIT_EVENT ——→ a timeout can be specified
- GET_ EVENT _ID
- GET_ EVENT _STATUS

Inter-partition communication

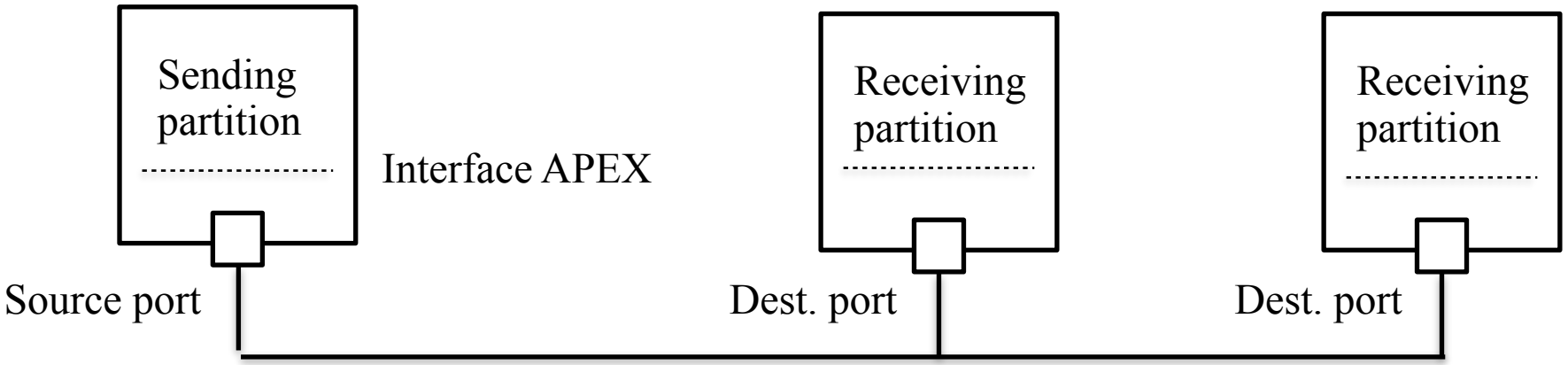
Communication between partitions (of the same module or not) is only done by message exchange.

A transmitter is connected to one or more receivers by a logical channel called "Channel"

The transmitting or receiving entities correspond to the partitions

A partition accesses a channel via a "Port

=> APEX service for sending and receiving via ports



Inter-partition communication

- The transmitting partition does not know the name and location of the receiving partitions.
- The physical connection between the sending and receiving ports is established by static network configuration.
- The communication protocol between transmitter and receiver must be the same.

  (same message format, same acknowledgement policy...)
- The periodic call of an APEX transmission service generates a periodic message, and conversely, an aperiodic call generates an aperiodic message.

  => the periodicity is not an attribute of the port

Inter-partition communication

Transfert protocol:

- Sampling mode:
  - the message always conveys the same updated data
  - each occurrence of a message overwrites the previous one
  - messages are of fixed size
  - the messages are not segmentable by the OS

- Queuing mode:
  - the message can convey different data
  - occurrences are saved in a FIFO
  - messages can be of variable length
  - messages are segmentable by the OS

Inter-partition communication

Ports attributes

- Identifier (id): value returned when creating the port
- Name
- Transfer protocol: sampling or queuing
- Direction: input or outpout
- Maximum message size
- Port size
  - identical to the message size for sampling ports
  - FIFO depth for queuing ports
- Refresh period for sampling ports
  - allows to monitor the freshness of the last received message
- Queuing mode waiting policy:
  - FIFO or priority

Inter-partition communication

Sampling APEX services

CREATE_ SAMPLING_PORT

WRITE_ SAMPLING_MESSAGE

READ_ SAMPLING_MESSAGE

reads the last message received in the port and indicates if the age of the message is consistent with the Refresh period attribute of the port

GET_ SAMPLING _ID

GET_ SAMPLING _STATUS

Service APEX queuing

CREATE_ QUEUING_PORT

SEND_ QUEUING _MESSAGE

RECEIVE_ QUEUING _MESSAGE ——→ a timeout can be specified

GET_ QUEUING _ID

GET_ QUEUING _STATUS

A conclusion

Summary:

- A two-level OS:
  - partition + static cyclic scheduling
  - process + dynamic scheduling by priority

- Intuitive objective: emulate classic federated architectures where each function has its own resources and manages its own processes as it sees fit

- Partition level required for certification requirements
  => guarantee that a faulty function will never be able to disrupt another one
  => notion of strict partitioning

Question: Wouldn't it have been simpler to remove the partition level and multiply the physical resources?

=> Non-Integrated Modular Avionics

=> Only resource sharing = the network