H4202 – PLANCHE Benjamin & PHAN Nicolas

TP Système d'Exploitation Avancée

Conception d'un pilote

Document de conception

Contenu

Architecture générale

Afin de remplir les spécifications préalablement établies, notre pilote repose sur une architecture employant diverses tâches et structures de données. A l'aide du langage de conception graphique LA4, nous allons dans un premier temps établir une vue générale de ce pilote.

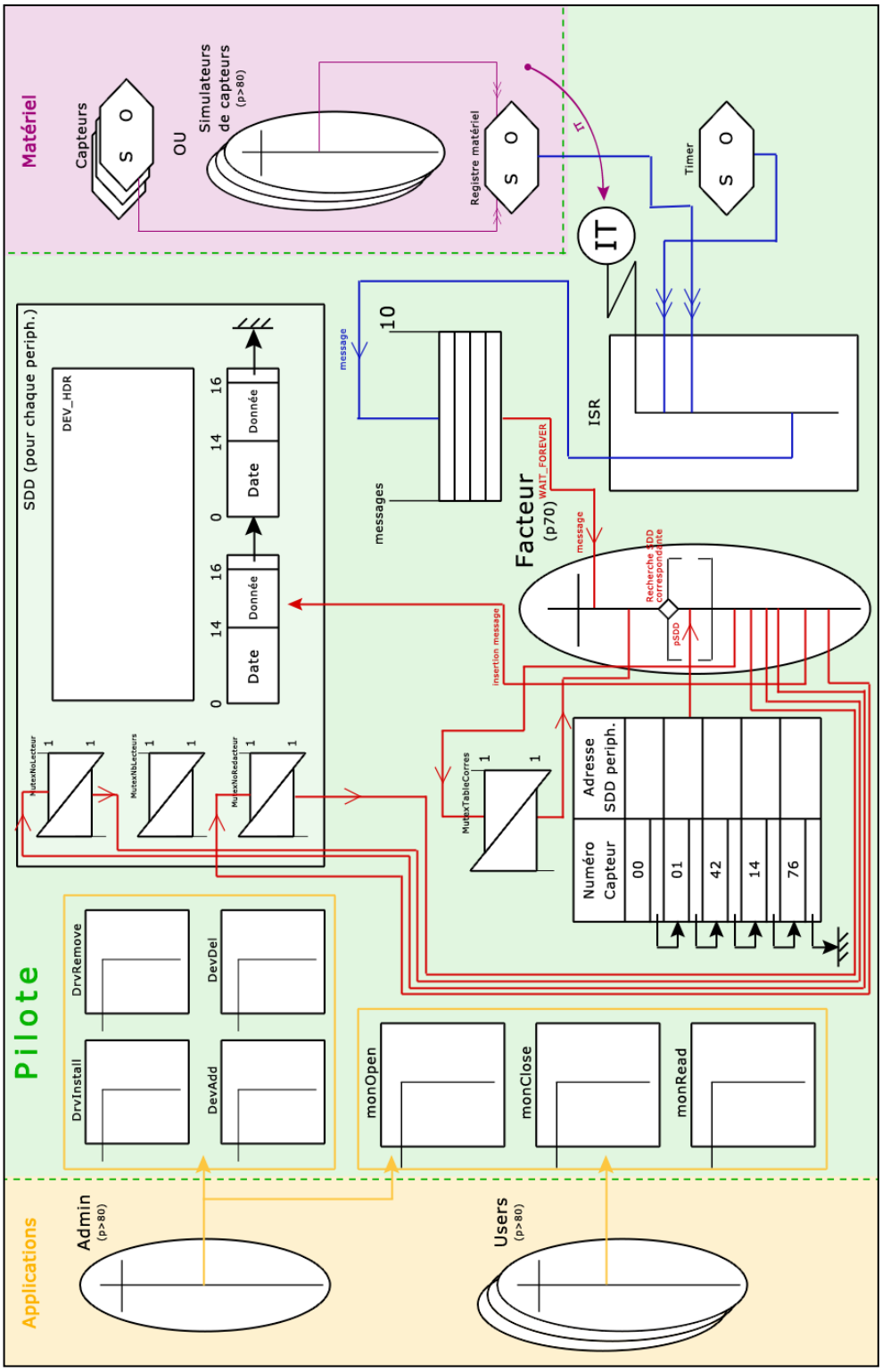

Schéma LA4 de l'architecture générale du pilote

Remarque :

En annexe se trouvent des schémas supplémentaires, explicitant les fonctions les plus complexes du pilote.

Spécification de l'ISR

Cette procédure correspond au *handler* associé à l'interruption matérielle générée par les capteurs désirant communiquer un message.

Chaque message envoyé par un périphérique a une taille fixe de 4 octets : la première partie du message, sur 2 octets, contient le numéro du capteur, et la deuxième partie contient la donnée du message (dans notre cas le numéro du badge qui a été passé devant ce capteur).

Lorsqu'un capteur a un message à transmettre, il stocke celui-ci dans un registre matériel commun à tous les capteurs, puis lance l'interruption qui activera l'ISR. Ininterruptible, celle-ci pourra sans risque traiter l'échange, le plus rapidement possible donc, avant de redonner la main.

Une des priorités de l'ISR est donc d'horodater le message, afin d'avoir une précision proche de la milliseconde. Pour se faire, elle devra donc consulter l'horloge interne du système, préalablement configurée.

La seconde priorité de cette procédure est de transmettre le contenu de l'échange au pilote et de *rendre la main* (pour traiter au plus vite les potentiels échanges suivants). Il est donc primordial que l'ISR contiennent le moins d'opérations possibles. Pour ce faire, l'ISR se contentera de transmettre le message à une boîte aux lettres. Celui-ci sera ensuite traité par une tâche propre au pilote, nommée « *Facteur* ».

L'ISR ne prend pas de paramètre. Elle a cependant accès au registre matériel des capteurs (dans notre cas, représenté par une variable globale).

De plus, elle ne renvoie aucun code d'erreur et ne demande aucun contrat d'utilisation : n'étant employée que dans un cas bien précis et n'étant pas interrompu, elle n'a à traiter aucun cas limite.

Spécification du facteur

Description

Cette tâche interne au pilote a pour fonction de récupérer les messages transmis par l'ISR dans la boîte aux lettres, et de les traiter « à son rythme », d'étudier le contenu du message pour en déduire le périphérique émetteur et ainsi sauvegarder le message dans la structure de données correspondante.

Cette tâche est initialisée à l'installation du périphérique, et tourne en boucle jusqu'à la désinstallation où elle sera arrêtée.

A chaque cycle, elle reste en attente illimitée sur la boîte aux lettres. Une fois un message horodaté récupéré, elle isole l'identifiant du capteur, puis parcourt la table de correspondance afin de trouver l'adresse de la structure de données associée au

périphérique correspondant au numéro. Le message, dépourvu de l'identifiant (donc date + numéro de badge) est alors copié en tête de pile, dans la liste des messages du périphériques.

Si la pile dépasse le nombre maximum de messages défini, *Facteur* supprime le message en queue.

Paramètres

Facteur prend en paramètre à son lancement l'identifiant de la boîte aux lettres, ainsi que l'identifiant du sémaphore protégeant la table de correspondance, afin de pouvoir les consulter.

Elle ne renvoie aucune donnée.

Cette tâche appartenant à la couche profonde du pilote, son protocole d'utilisation et son environnement (boîte aux lettres, table de correspondance) sont clairement définis : toute erreur sur les données sera détectée en amont (notamment par les fonctions d'installation du pilote et des périphériques). *Facteur* n'a donc pas à traiter de cas limites, et à renvoyer de codes-retour.

Priorité

Tâche motrice du pilote, *Facteur* possède une priorité élevée, de 70. Cette valeur est censée lui permettre de passer avant la plupart des tâches applicatives définies dans notre cas.

Spécification de la SDD

En plus des données système requis pour la gestion d'un périphérique, notre pilote requiert des valeurs particulières, liées à notre cas d'utilisation. En voici la liste :

- ***debutListe***, de type Message*
  - Début de la liste chaînée de messages mémorisés pour ce périphérique.
  - La structure Message contient les champs :
    - ***donnee***, de type char[2]
      - Teneur du message (numéro de badge) sur 2 octets.
    - ***date***, de type char[*MSG_DATE_LENGTH*]
      - Horodatage à la milliseconde, au format : hh:mm:ss-xxx
    - ***suivant***, de type struct ElemMessage*
      - Pointeur vers le message suivant (message plus ancien).
- ***nbMessage***, de type int
  - Nombre de messages actuellement contenu dans la pile de ce périphérique.
- ***IdSemNoLecteurs***, de type SEM_ID
  - ID du sémaphore FIFO destiné aux lecteurs de messages sur ce périphérique.
- ***idSemNoRedacteur***, de type SEM_ID
  - ID du sémaphore FIFO destiné au rédacteur.
- ***idSemNbLect***, de type SEM_ID

- ID du sémaphore FIFO destiné à protéger *nbLect*.

- ***nbLect***, de type int
  - Nombre de lecteurs sur les messages du périphérique.

Remarque :

- Ces variables et la structure *DEV_HDR* sont contenus comme SDD sous le type nommé ***DEV***.

- Pour le stockage des messages, l'emploi d'une liste chaînée a été privilégié à celui d'un tableau, pour la gestion dynamique qu'offre la première. Le coût supplémentaire que cela ajoute au parcours des messages est négligeable, dans le sens où la lecture sera probablement la plupart du temps restreinte aux messages les plus récents, en sommet de pile. L'avantage vient dans la gestion du nombre de messages à sauvegarder. Nous pouvons imaginer l'implémentation d'une fonctionnalité ioctl permettant de modifier ce nombre, action facilitée par la gestion par liste.

- La variable *nbLect* et les trois sémaphores associés à chaque périphérique font partis d'un ensemble, permettant la protection de la pile de message par un système Lecteurs-Rédacteur. Celui-ci permet à plusieurs lecteurs de consulter simultanément les messages en excluant toute possibilité de modification. Lorsque *Facteur* veut modifier la liste, ce « rédacteur » s'insère dans la liste d'attente des tâches lectrices, attendant que toutes les tâches mieux classées aient fini leur consultation. Une fois seul face aux messages (zéro lecteur), le rédacteur peut alors éditer, excluant toute possibilité de lecture.

  - *idSemNoRedacteur* sert à l'exclusion des lecteurs lorsqu'une rédaction est en cours.
  - *IdSemNoLecteurs* sert inversement à l'exclusion du rédacteur lorsqu'une lecture est en cours.
  -
  - *idSemNbLect* protège la variable *nbLect*, incrémentée à chaque arrivée de lecteur et décrémentée à chaque départ.
    - Lorsque le premier lecteur arrive (*nbLect* précédemment à 0), il prélève le jeton du sémaphore *NoLecteurs*.
    - Lorsque le dernier lecteur s'en va (*nbLect* précédemment à 1), il rend ce jeton, rendant possible l'édition.

Variables globales & éléments de synchronisation.

Plusieurs variables globales et éléments de synchronisation sont employés au fonctionnement du pilote, internes ou non à celui-ci.

- ***tableCorrespondance***, de type TableCorrespondance
  - Cette liste permet de faire l'association entre un numéro matériel de périphérique et l'adresse de sa SDD. Mise à jour à chaque ajout et suppression de périphérique, elle permet ainsi à *Facteur* de retrouver la SDD dans lequel enregistrer un message à partir de sa signature.
  - Cette structure contient :
    - ***numCapteur***, de type char[2]
      - Identifiant numérique du capteur.
    - ***pointeurSDD*** , de type DEV*
      - Pointeur vers la SDD du capteur.
    - ***suivant***, de type struct TableCorrespondance*
      - Pointeur permettant le chaînage de la liste.
  - Elle est créée à l'installation du pilote, par *DrvInstall*, et est détruite avec lui, par *DrvRemove*.

- ***Mutex*** de tableCorrespondance
  - Mutex basé sur la priorité protégeant les accès à la table de correspondance (lecture par Facteur, édition par les fonctions *DevAdd* et *DevDel*).
  - Son identifiant se nomme *idSemTableCorrespondance*, de type SEM_ID.
  - Il est créé à l'installation du pilote, par *DrvInstall*, et est détruit avec lui, par *DrvRemove*.

- ***Boîte aux lettres***
  - Seule boîte aux lettres du pilote, elle sert à stocker les messages reçus en attente de traitement par *Facteur*.
  - Elle peut contenir jusqu'à 10 messages de taille normée (MAX_MSG_LENGTH = 12).
  - Son identifiant se nomme *idBoiteAuxLettres*, de type MSG_Q_ID.
  - Elle est créée à l'installation du pilote, par *DrvInstall*, et est détruite avec lui, par *DrvRemove*.

- ***Registre***, de type char[2]
  - Variable externe au pilote, simulant le registre matériel utilisé par les capteurs pour stocker temporairement leur message.

Annexe.

Contient :

- Schéma LA4 de l'architecture générale (format pleine page)
- Schéma LA4 de la fonction *DrvInstall* (format pleine page)
- Schéma LA4 de la fonction *DrvRemove* (format pleine page)
- Schéma LA4 de la fonction *DevAdd* (format pleine page
- Schéma LA4 de la fonction *DevDel* (format pleine page)
- Schéma LA4 de la fonction *Read* (format pleine page)

Schéma LA4 de l'architecture générale du pilote

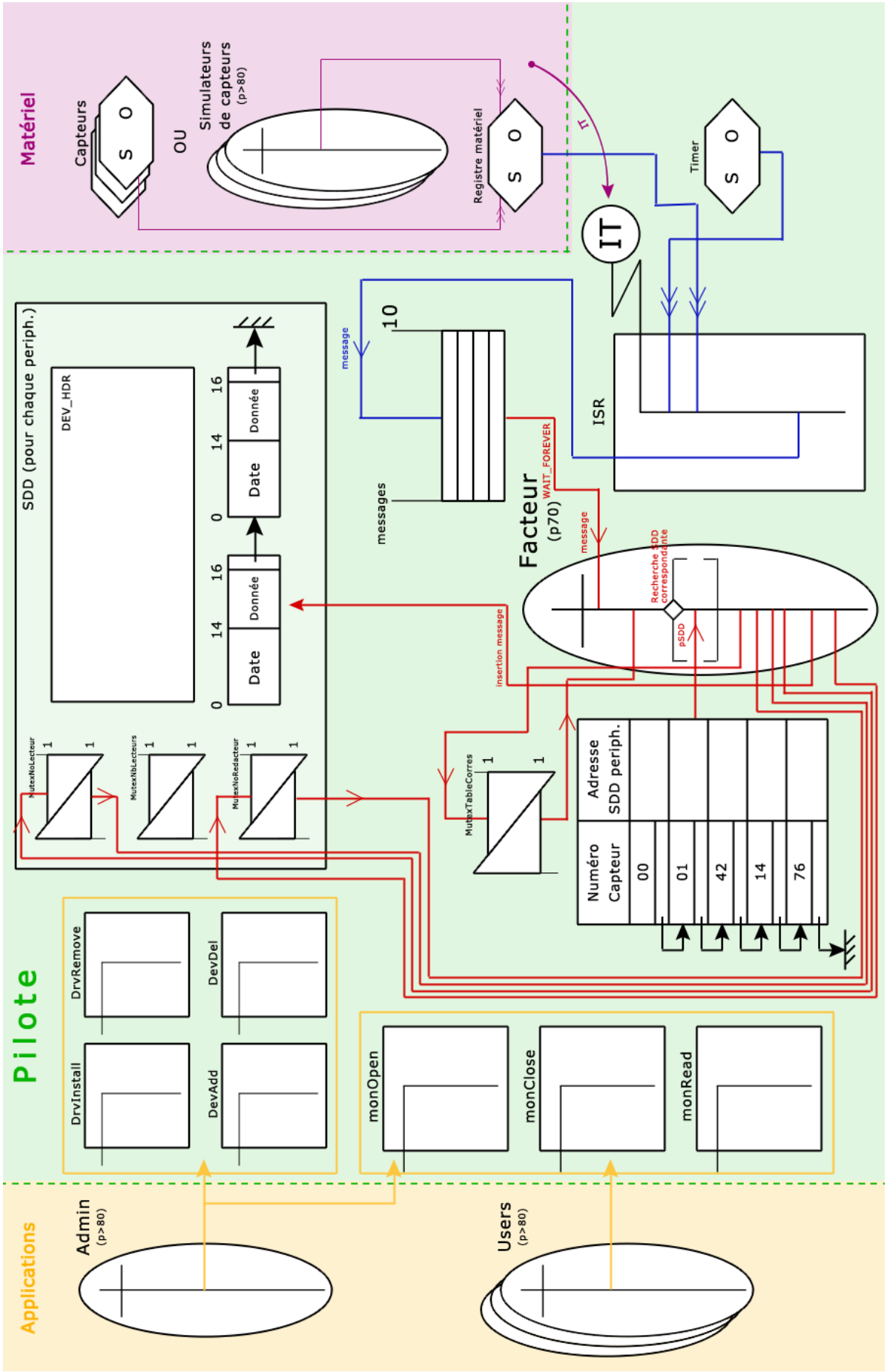

Schéma LA4 de la fonction *DrvInstall*

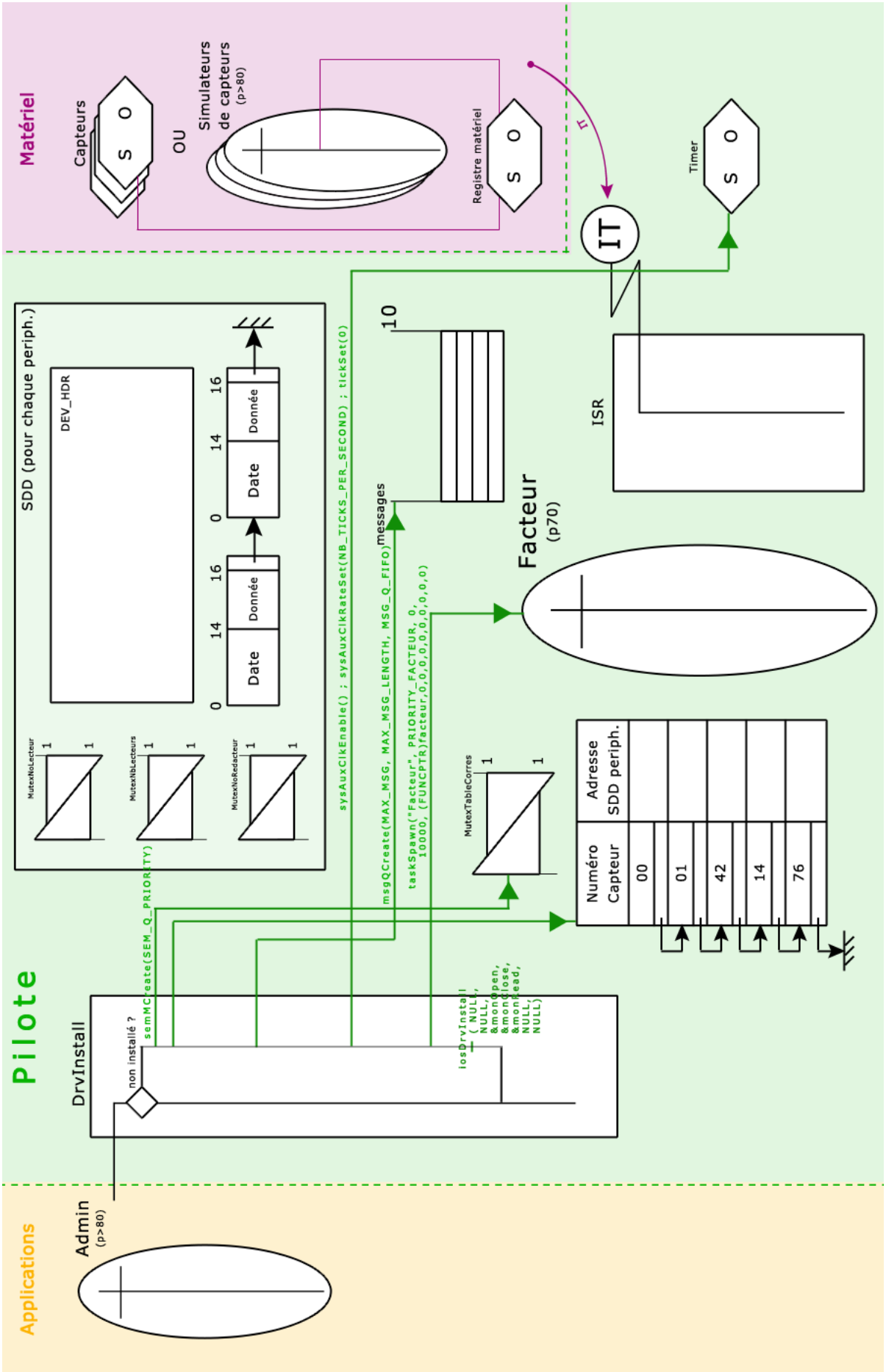

Schéma LA4 de la fonction *DrvRemove*

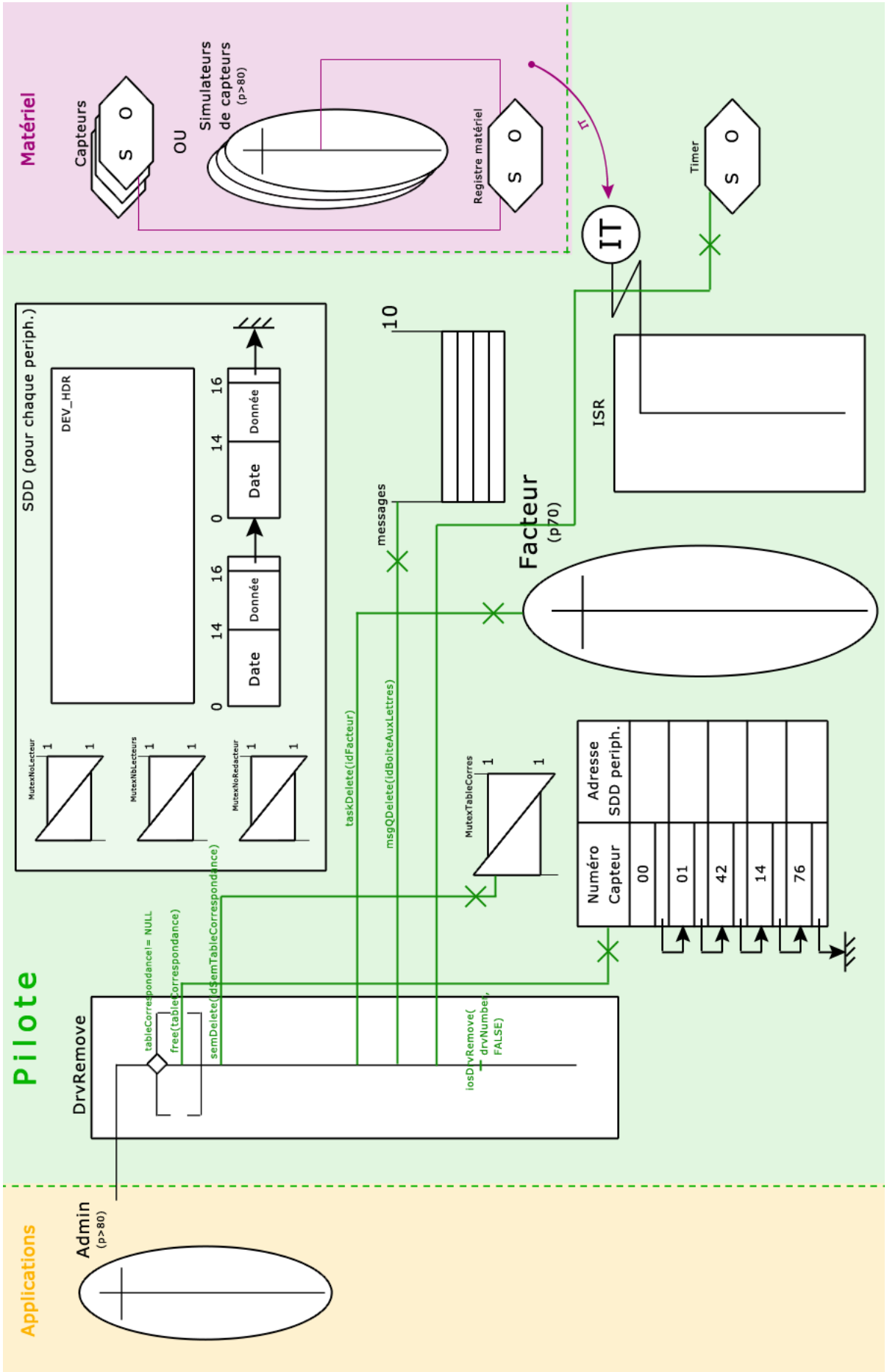

Schéma LA4 de la fonction *DevAdd*

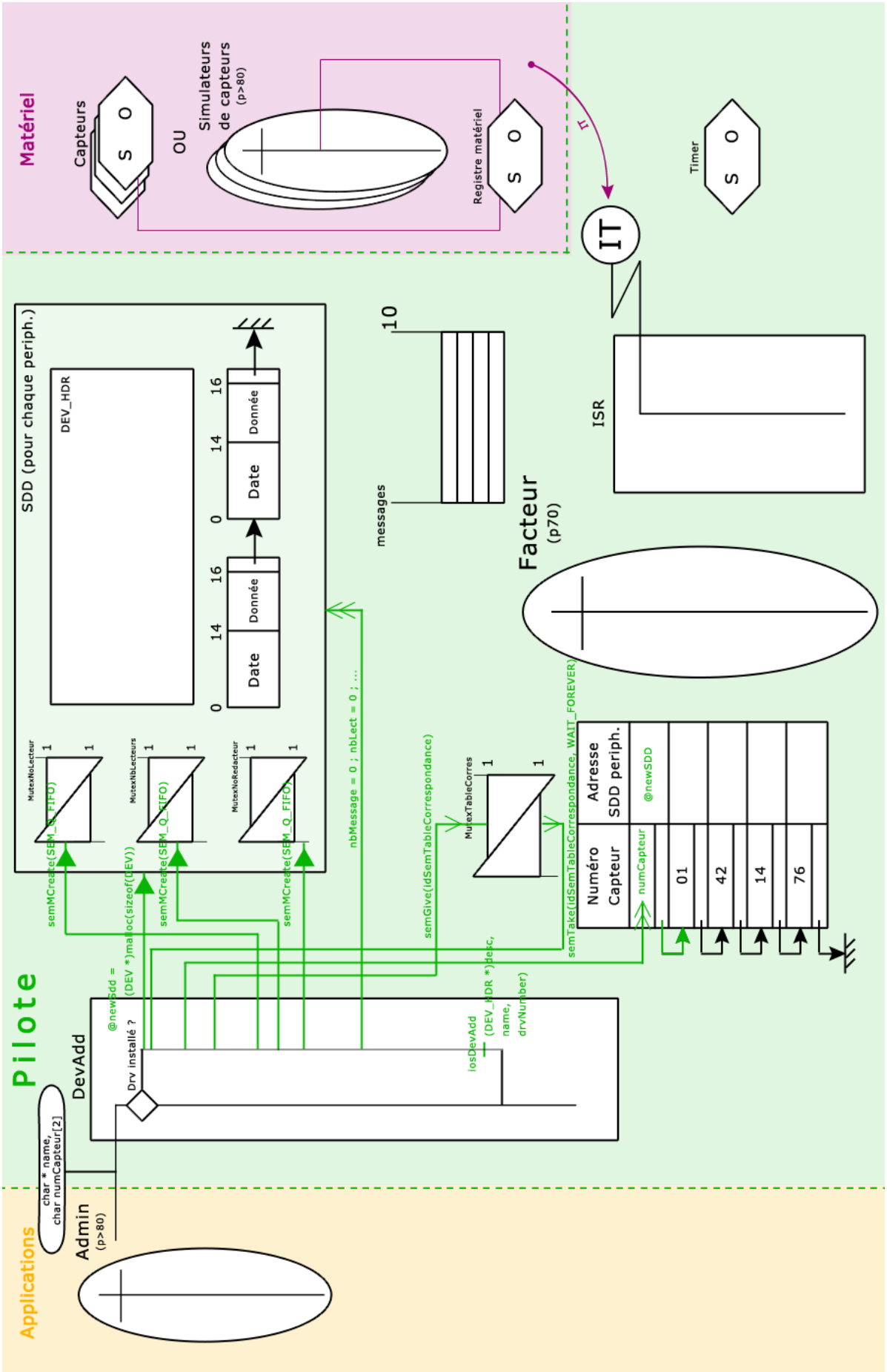

Schéma LA4 de la fonction *DevDel*

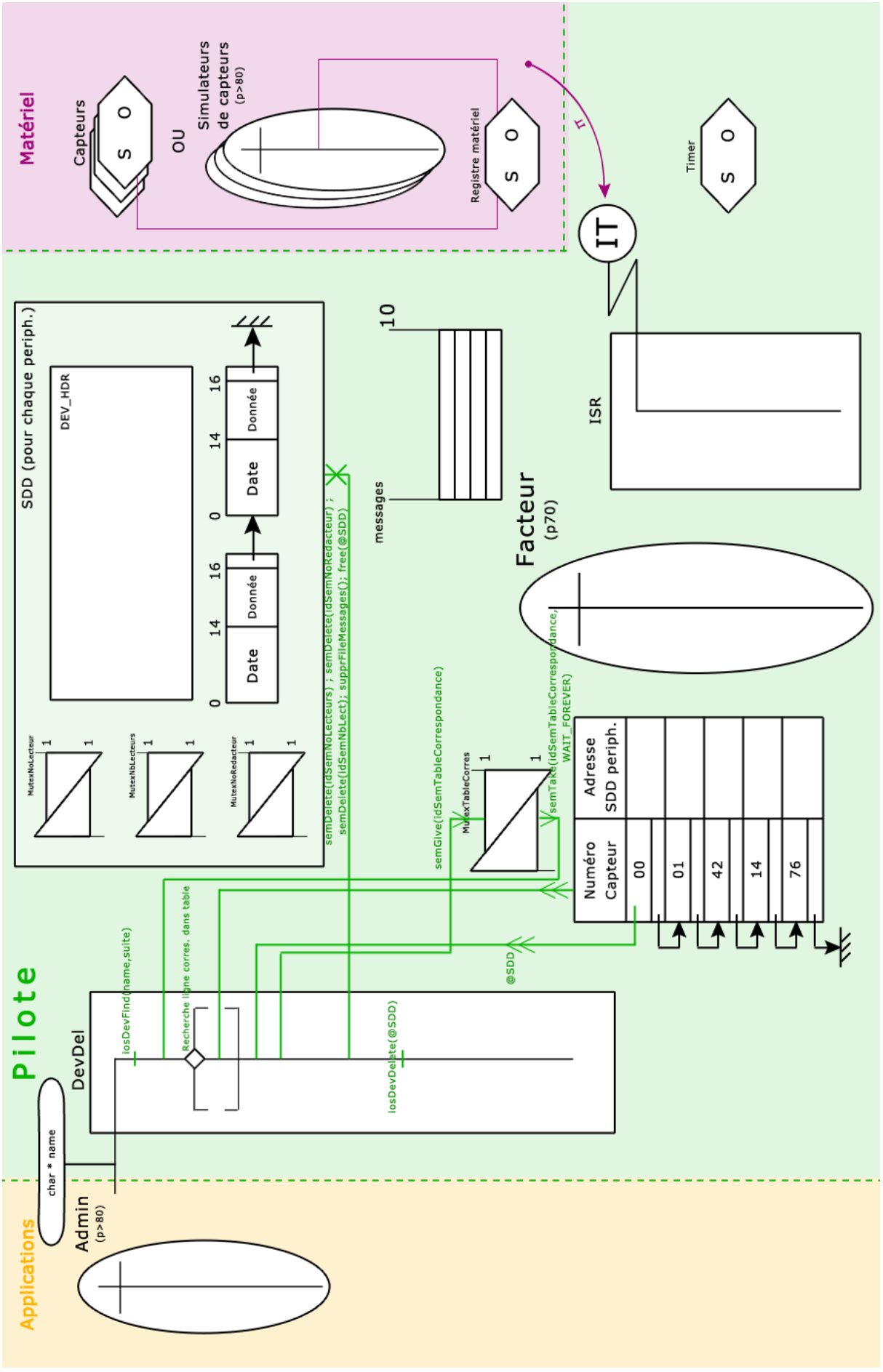

Schéma LA4 de la fonction *Read*