BTS INFORMATIQUE ET RESEAUX

POUR L'INDUSTRIE ET LES SERVICES TECHNIQUES

Session 2004

Epreuve E.4
Etude d'un Système Informatisé

Traitement de pièces en ABS par galvanoplastie

Sujet

Durée : 6h00 ***Coefficient* 5**

"Calculatrice autorisée (conformément à la circulaire n° 99-186 du 16 novembre 1999)."

Aucun document autorisé.

Ce document comprend 39 pages composées de :

Sujet	: pages 1 à 16	sur papier rose
Annexes	: pages 1 à 10	sur papier vert
Document réponse	: pages 1 à 13	sur papier blanc
	à rendre obligatoirement (même vierge) .	

Toutes les réponses sont à fournir sur le livret « Document Réponse » à l'exclusion de tout autre support.

Les réponses doivent être exclusivement fournies dans les emplacements prévus à cet effet. Si nécessaire le candidat a la possibilité de rectifier ses réponses sur la page non imprimée en regard.

On ne justifiera une réponse que si le document le demande.

Temps conseillés et barèmes indicatifs:

Lecture du sujet :	**30 minutes.**	
B) Analyse du système :	**60 minutes.**	**20 points**
C) Communication Bus CAN :	**60 minutes.**	**20 points**
D) Système temps réel :	**60 minutes.**	**20 points**
E) Programmation :	**60 minutes.**	**20 points**
F) Communication et réseau :	**60 minutes.**	**20 points**
Relecture :	**30 minutes.**	

A. Présentation du Système

A.1 Expression du besoin

La société X, sous-traitante de l'industrie automobile, fabrique des pièces en ABS (matière synthétique) métallisé (insignes, éléments de décoration, etc.).
Le procédé de fabrication (métallisation de pièces en ABS, apparenté à la galvanoplastie) fait appel à une unité de production développée par la société **TUBALEX**. Le sujet se place du point de vue d'un technicien IRIS en charge de la gestion et de la maintenance de la ligne de production.

A.2 Principe général

Le procédé se décompose en quatre grandes phases :

1. Prétraitement de la surface dans un bain d'acide sulfochromique (afin de rendre cette surface rugueuse).
2. Désoxydation de la surface,
3. Post-activation de la surface par dépôt électrolytique d'un film riche en cuivre, très conducteur,
4. Dépôt électrolytique de la couche de finition, en l'occurrence du chrome.

Chacune de ces phases demande l'immersion des pièces à traiter dans un bain, pour une durée déterminée, différente pour chaque phase. Ces différentes phases sont en général séparées par des cuves de rinçage.

Le nombre de bains identiques pour une phase donnée dépend de la durée relative de cette phase dans la chaîne. Chaque outillage n'est immergé que dans un bain par phase.

Fig. 1 : Photo de l'installation.

A.3 Détail de l'installation

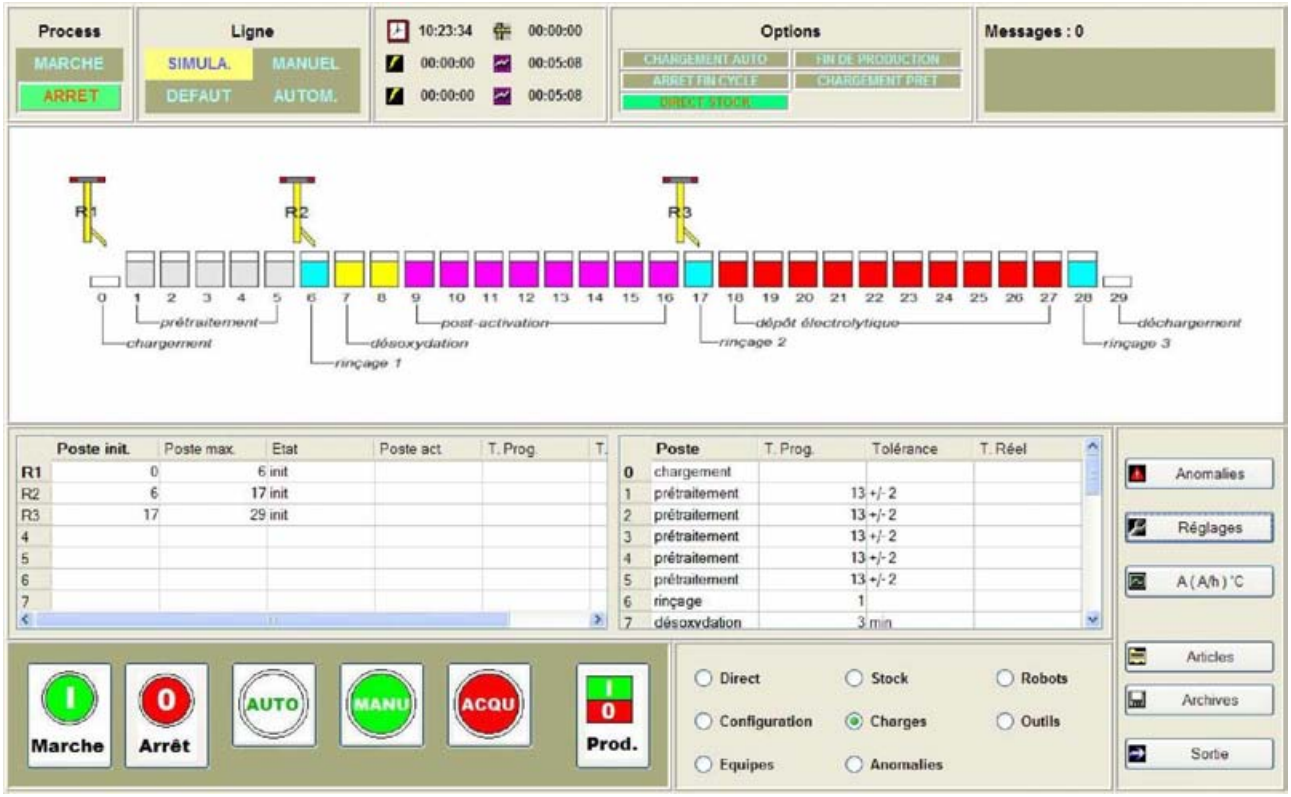


Fig. 2 : Capture d'écran du poste de contrôle/commande (pour information).

La ligne de fabrication à laquelle nous nous intéressons est composée de 28 bains alignés, numérotés de 1 à 28, précédés d'un poste de chargement de numéro 0 et suivis d'un poste de déchargement de numéro 29.
L'espace séparant un bain du bain suivant est de 50 centimètres. Chaque bain est équipé de capteurs et d'actionneurs permettant de

- Remplir/Vidanger/Vérifier les niveaux du bain,
- Chauffer/Refroidir les bains,
- Contrôler l'électrolyse,
- Contrôler la présence de l'outillage.

L'ensemble de ces capteurs/actionneurs est piloté par un boîtier d'Entrées/Sorties situé à proximité du bain.

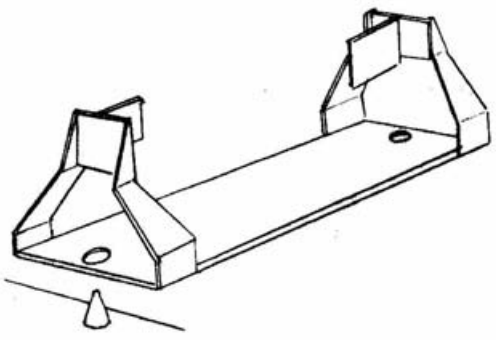


Fig. 3 : Détail d'un outillage.

Les pièces à métalliser sont fixées par grappes à un **outillage** (encore appelé **portant**) qui peut être déposé précisément sur un bain par le biais de plots de centrage.

Ces portants sont manipulés par des robots deux axes équipés d'une potence disposant d'un mécanisme de portage simple ne demandant aucune manipulation extérieure.

*Note : Par la suite on ne parlera plus que d'**outillage**.*

Ce mécanisme permet au robot d'immerger un outillage dans un bain, de le sortir, de l'égoutter et de le transporter d'un bain vers un autre.

La ligne de fabrication est équipée de trois robots (R1, R2 et R3). Chaque robot possède une zone d'influence limitée lui permettant de desservir plusieurs bains, et présentant une intersection d'un bain avec les robots voisins.

Chaque robot est équipé de capteurs et d'actionneurs permettant le mouvement de la potence sur la ligne (horizontalement et verticalement) . Ces capteurs et actionneurs sont pilotés par le biais d'un module CAN fixé sur le robot…

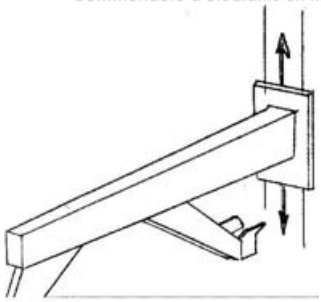


Fig. 4 : Détail d'une potence.

A.4 Synoptique général

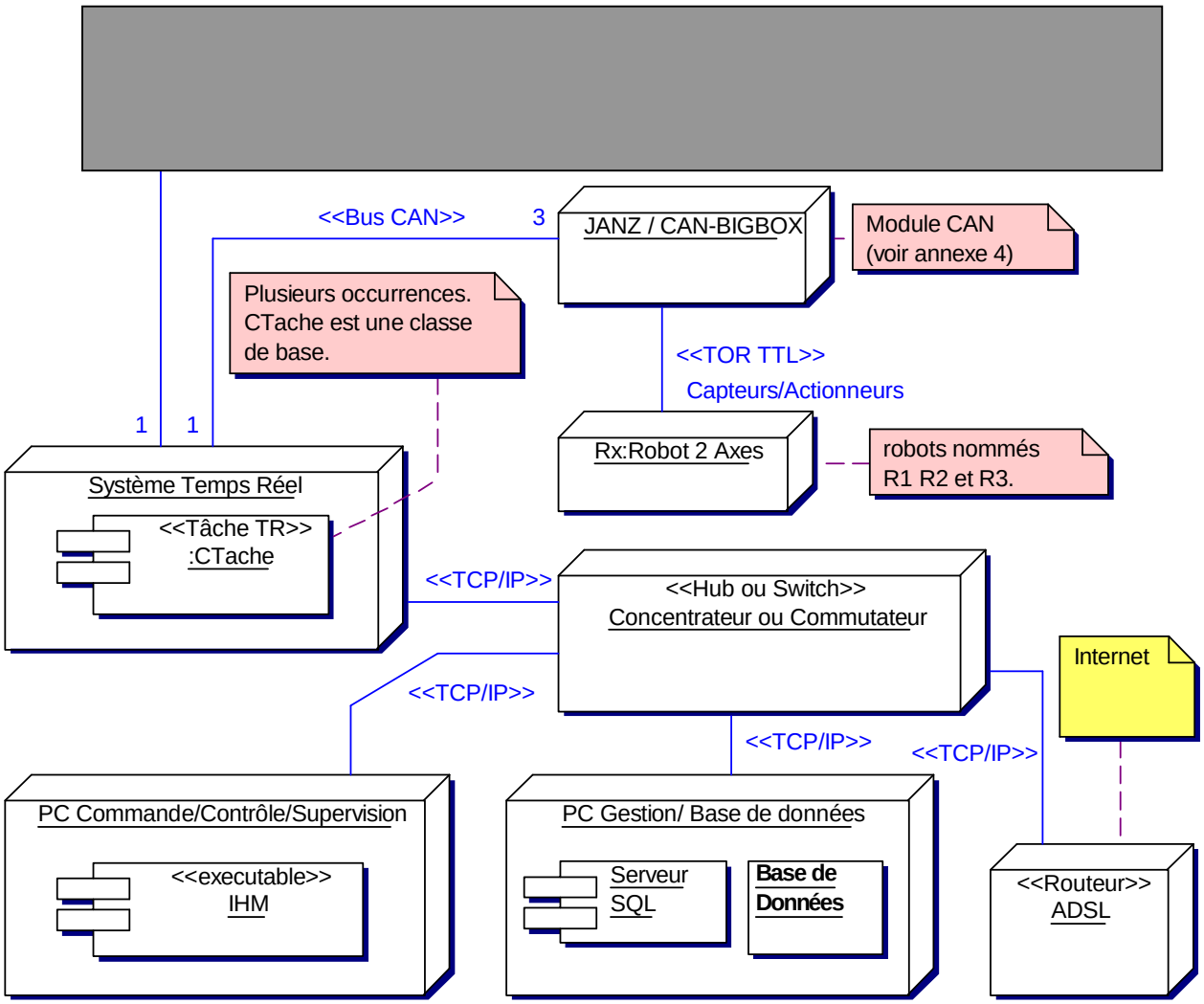


Fig. 5 : Diagramme de déploiement.

B. Analyse du Système

B.1 Répartition des bains

Les 30 postes (28 bains + chargement et déchargement) sont répartis de la manière suivante :

N° du poste	Phase	Durée en minutes	Remarques
0	Chargement		
1-5	Prétraitement	13'±2'	*5 postes identiques (traitement //)*
6	Rinçage 1	1'	***Zone de partage entre R1 et R2***
7-8	Désoxydation	3'	*2 postes identiques (traitement //)*
9-16	Post-activation	21'±2'	*8 postes identiques (traitement //)*
17	Rinçage 2	1'	***Zone de partage entre R2 et R3***
18-27	Dépôt électrolytique	28'±3'	*10 postes identiques (traitement //)*
28	Rinçage 3	1'	
29	Déchargement		

Rappel : Chaque outillage n'est immergé que dans un bain par phase.

On appelle **transfert** l'action consistant à déplacer un outillage d'un poste vers un autre poste.

Chaque transfert est composé de la séquence suivante (voir annexe 2):

- Déplacement horizontal du robot vers le poste source.
- Déplacement vertical du robot vers l'outillage (descente).
- Accrochage de l'outillage.
- Déplacement vertical pour sortie de bain de l'outillage (montée et égouttage).
- Déplacement vertical de dégagement du robot (montée).
- Déplacement horizontal du robot vers le poste destination.
- Déplacement vertical du robot vers le bain (descente).
- Décrochage de l'outillage.
- Déplacement vertical de dégagement du robot (montée).

L'ensemble de ces opérations requiert une durée de 1 minute. Seuls les transferts au niveau des postes de chargement et de déchargement ne respectent pas exactement ces opérations.

Question B.1.1
En regardant le tableau ci-dessus, déterminer le nombre de transferts que devra subir un outillage entre le poste de chargement et le poste de déchargement.

Question B.1.2
Calculer les durées théoriques minimale et nominale nécessaires à la production d'une pièce sur la chaîne, hors phase de chargement et de déchargement.

En entrée de la chaîne, un outillage avec ses grappes de pièces à métalliser est disponible toutes les 3 minutes.

Question B.1.3
Justifier numériquement le choix du nombre de bains pour chacune des différentes phases, hors phase de chargement et de déchargement.

B.2 Mouvement du robot R1

Le robot R1 a la particularité d'être responsable du poste de chargement. La mise en place de l'outillage se fait manuellement, et bien que l'opération soit simple et ne nécessite que quelques secondes, on souhaite que le robot passe le plus de temps possible au poste de chargement afin de permettre aux opérateurs de faire leur travail dans les meilleures conditions possibles.

Question B. 2.1
Etablir le nombre des transferts que doit effectuer le robot R1 pour un outillage donné. Préciser succinctement ces transferts.

On s'intéresse maintenant au déplacement horizontal du robot R1 entre le poste 0 et l'un des postes 1 à 5. L'accélération du robot sur cet axe est supposée constante et égale à 2 m/s². La décélération est de 8 m/s². La vitesse maximale est de 1 m/s, une vitesse plus faible (0,2 m/s) étant engagée à l'approche du poste à desservir. (voir Annexe 7 : Formulaire de mécanique)

Question B.2.2
Calculer le temps nécessaire au robot, partant de l'arrêt complet, pour atteindre sa vitesse maximale.

Question B.2.3
Calculer la distance parcourue par le robot pendant cette phase d'accélération.

Question B.2.4
Calculer la distance parcourue pour passer de la vitesse maximale à la vitesse faible.

Question B.2.5
Calculer la distance parcourue par le robot pour passer de la vitesse faible à l'arrêt.

Des capteurs de type inductif sont placés sur les robots afin de détecter des drapeaux métalliques situés dans l'axe du poste pour l'arrêt, ainsi qu'à dix centimètres de part et d'autre de l'axe du poste. Les marqueurs sont nommés G_n, C_n et D_n avec n numéro du bain (voir annexe 1).

Question B.2.6
Quel est le principe de fonctionnement d'un capteur inductif ?

Question B.2.7
Quel est l'intérêt de ce type de capteur dans l'environnement industriel étudié ?

C. Communication : Bus CAN

Les divers capteurs et actionneurs de chaque robot sont reliés aux entrées/sorties d'un module CAN-BIGBOX fabriqué par la société JANZ (voir Annexe 1 et Annexe 6).

C.1 Commande des moteurs et Lecture des capteurs

Pour lire les entrées, il faut demander au module CAN-BIGBOX via le bus CAN d'effectuer une lecture à l'adresse $1000 (1000 en hexadécimal) ; pour piloter un actionneur, de réaliser une écriture à l'adresse $1001.

Question C.1.1
Le capteur Z d'un robot est représenté par le bit de poids faible de l'octet situé à l'adresse $1000 (une valeur logique 1 indique que le robot est en position d'initialisation). Proposer une opération logique (opérateur et opérande) afin d'isoler ce bit dans l'octet lu.

L'octet permettant de piloter les actionneurs d'un robot est organisé comme suit :

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Marche(1) Arrêt (0)	X	Rotation : Horaire (1) Trigo (0)	X	Vitesse : Grande (1) Petite (0)	X	X	X

Note 1 : un arrêt n'est pas obligatoire lors d'un changement de vitesse
*Note 2 : **X** = Bit non utilisé. Valeur indifférente.*

Question C.1.2
Quelles valeurs hexadécimales doit prendre successivement cet octet pour ordonner un déplacement en petite vitesse puis en grande vitesse dans le sens horaire ?

Question C.1.3
Vérifier l'adéquation de ce module aux besoins de l'application. Indiquer pour quelles raisons les autres modules proposés n'ont pas été retenus. (voir annexe 4 - Equipements : documents constructeurs)

C.2 Dépose et Reprise d'outillage

UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation, à l'aide de diagrammes d'activités (une variante des diagrammes d'état-transitions).

Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles. Le passage d'une activité vers une autre est matérialisé par une transition qui peut être conditionnelle ou automatique.

Les transitions sont déclenchées par la fin d'une activité et provoquent le début d'une autre activité.

Question C.2
On fournit en annexe 3 le diagramme d'activité décrivant les opérations élémentaires nécessaires à la prise d'un outillage. Réaliser sur le même modèle et en vous aidant de l'annexe 2, le diagramme d'activités lié à la dépose d'un outillage.

C.3 Protocole CAN

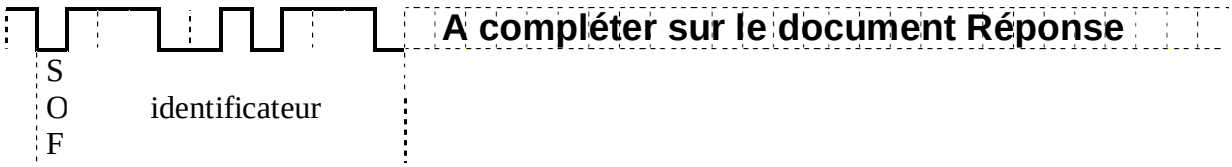
Le dialogue entre le système temps réel et un module CAN (ordre ou information capteur) est basé sur des trames qui comportent un champ de données de 16 bits exactement (voir annexe 1).

Question C.3.1
En vous référant à la documentation du bus CAN fournie en annexe 5 et aux informations ci-dessus, déterminer la longueur L en bits d'une trame de données circulant sur le bus (on ne tient pas compte des problèmes de *bit-stuffing*).

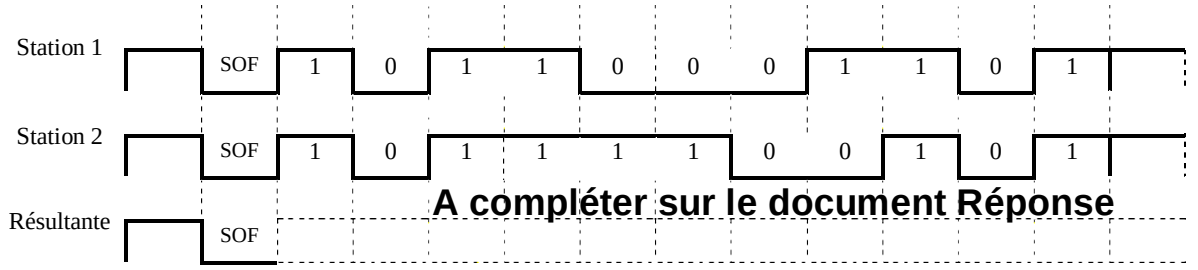
Question C.3.2
En déduire le nombre maximum de trames pouvant circuler sur le bus en une seconde.

Question C.3.3
Dans le cas le plus défavorable d'un déplacement horizontal entre des bains non voisins, chacun des trois modules CAN affectés aux robots émet et reçoit un maximum de 8 trames par seconde. Déterminer le taux de charge maximum du bus.

Question C.3.4
On donne le SOF et la partie indentificateur du champ d'arbitrage d'une trame contenant l'ordre « A\006 » (Caractère ASCII 'A' suivi de l'octet de valeur 6). Compléter la trame jusqu'au champ CRC exclu. *(on ne tient pas compte de la règle du « stuffing »)*



Question C.3.5
Deux stations désirent émettre une trame dont les chronogrammes sont donnés ci-après. Dans le cas où ces trames seraient émises en même temps, le mécanisme d'arbitrage du bus CAN va résoudre le conflit (voir annexe 5). Compléter le troisième chronogramme (résultante sur le bus) et indiquer quelle station a réussi à émettre sa trame.



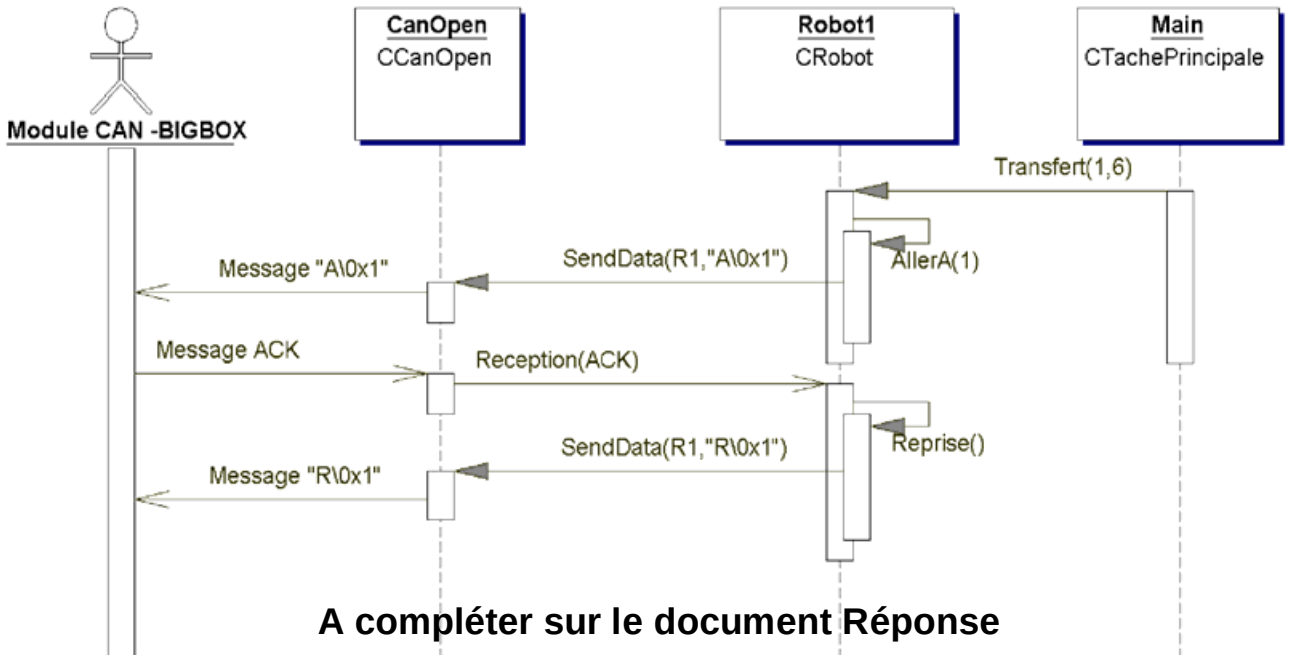
D. Système temps réel

Le système temps réel prend en charge la gestion de tous les modules d'entrée-sortie reliés au bus CAN. Il est lui-même équipé d'une carte CAN qui sera gérée par un objet de type CCanOpen.

CCanOpen
… …
+ SendData(int Id, char* Msg) : void + OnReceive(int Id, char* Msg) : void … …

Nota: *OnReceive est appelée de manière asynchrone sur réception d'une trame sur le bus CAN.*

Question D.1
Complétez le diagramme de séquence ci-dessous, qui décrit une opération de transfert d'un outillage entre les bains 1 et 6…



Nota : *Précision sur les messages (Notation UML 1.4)*

Message synchrone (appel de fonction ou de méthode).
Message asynchrone (lié à un évènement ou à une interruption).

D.2 Analyse d'une situation de panne

A la suite d'un incident, le câble qui relie le capteur horizontal du robot R1 au module CAN-BIGBOX est rompu. Le robot est en position haute.
Le système temps réel émet l'ordre A(1) (Aller au poste 1) vers le module CAN…

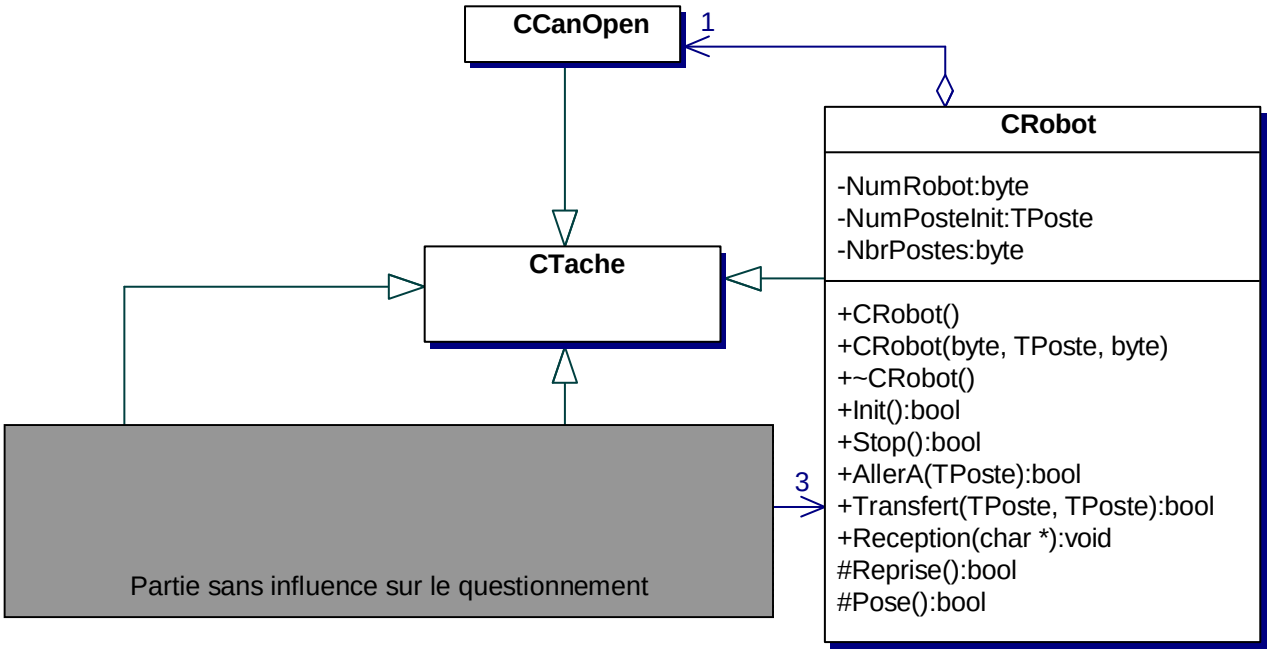
Question D.2.1
En l'absence de tout mécanisme de sécurité, que risque-t-il de se passer ?

Question D.2.2
De quelles informations et grandeurs physiques (citées en B.2) doit disposer le système temps réel pour déterminer le délai sous lequel il doit recevoir une information capteur en provenance du module CAN ?

Question D.2.3
Quel mécanisme logiciel des systèmes multi-tâches ou temps réel peut-on mettre en œuvre pour que la tâche en charge de la gestion du robot R1 soit réveillée à la suite de cet incident ?

Question D.2.4
Après ce réveil, quelle décision doit prendre la tâche en question ?

E. Programmation



E.1 Modélisation de la classe CRobot

Question E.1.1
Proposez une déclaration en langage C++ de la classe CRobot conforme au diagramme ci-dessus.

Pour des raisons de maintenance du système (analyse de défaillances, vérification des temps de cycle, …), il est nécessaire de conserver, au niveau de chaque instance de la classe CRobot, un historique d'activité regroupant les informations datées suivantes : ordres reçus, réponses transmises, capteurs détectés.

Codage d'une information d'ordre :
- un caractère parmi 'I', 'S', 'A', 'P', 'R' pour Init(), Stop(), AllerA(), Pose(), Reprise()
- un numéro de poste (mis par convention à 0 dans le cas des ordres 'I' et 'S')

Codage d'une information capteur (position):
- un identifiant parmi 'Z', 'H', 'B', 'E', 'X', 'G', 'C', 'D'
- un numéro de poste

Codage d'une information de réponse :
- POM ou ACK ou NAK

Question E.1.2
Proposez une structure de données de type TOrdre assurant le codage d'une information d'ordre.

Question E.1.3
De la même manière, donnez une définition du type TPosition assurant le codage d'une information capteur.

Question E.1.4
Les réponses possibles émises par le robot sont implémentées comme suit :
```
enum TReponse { POM = 0xFD, ACK, NAK } ;
```
Commentez cette ligne de déclaration.

On dispose d'un type structuré de données nommé T_timestamp permettant le codage d'une date et d'une heure avec une précision de la milliseconde.

Question E.1.5
Proposez la définition du type TInfo représentant un élément de l'historique. Cet élément doit regrouper une datation `date` suivie d'une **union** pouvant être une information d'ordre, de capteur ou de réponse.

E.2 Gestion de l'historique

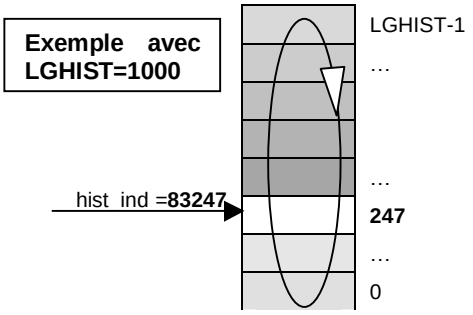
Les membres privés qui suivent sont ajoutés à la classe CRobot :
```
TInfo          hist[LGHIST] ;
unsigned int   hist_ind ;
```

Le tableau **hist** est rempli avec des octets nuls par le constructeur, **hist_ind** est initialisé à 0.

L'historique est en réalité un tampon circulaire assurant le stockage des LGHIST informations les plus récentes.

L'itérateur hist_ind :

- permet le comptage des informations depuis la naissance de l'objet CRobot,
- donne accès en permanence à l'emplacement suivant la dernière information enregistrée.



On dispose de la fonction `T_timestamp& GetTimestamp()` assurant la lecture instantanée des date et heure du système.

Question E.2.1
Les extraits de code suivants proposent des solutions d'enregistrement d'une datation. Pour chaque extrait de code, préciser s'il répond aux besoins, justifiez en cas de réponse négative.

solution 1 :
```
hist[hist_ind++].date = GetTimestamp() ;
```

solution 2 :
```
if ( hist_ind >= LGHIST )     hist_ind = 0 ;
hist[hist_ind++].date = GetTimestamp() ;
```

solution 3 :
```
hist[hist_ind].date = GetTimestamp() ;
if ( hist_ind++ > LGHIST )    hist_ind = 0 ;
```

solution 4 :
```
hist[ hist_ind++ % LGHIST ].date = GetTimestamp() ;
```

Question E.2.2
Proposez le développement d'une nouvelle méthode privée permettant l'inscription dans l'historique d'une information de type ordre. Cette méthode doit répondre au prototype suivant :
`void CRobot::AjoutHisto(TOrdre& ordre) ;`

Question E.2.3
On souhaite, de la même manière, pouvoir inscrire les informations de type position et réponse dans l'historique. Proposez des prototypes de surcharge de la méthode précédente afin de satisfaire à ce besoin.

Les informations maintenues dans l'historique sont susceptibles d'être remontées vers le système de supervision. La classe CRobot dispose en réalité pour cela d'un jeu de méthodes publiques permettant aux objets clients d'interroger son historique.

Question E.2.4
Développez en quelques lignes la méthode `TOrdre CRobot::DernierOrdre()` permettant la récupération de la dernière information de type TOrdre datée dans l'historique (hors de tout mécanisme lié au temps réel).

F. Communication et réseau

F.1 Organisation du réseau

Le réseau mis en œuvre est à base d'une architecture Ethernet 100baseT. La communication entre le système temps réel et le système de supervision utilise un protocole propriétaire.

Question F.1.1
Compléter le tableau en donnant le numéro et le nom de la couche du modèle OSI concernée par les différentes entités ou protocoles présents sur le réseau utilisé.

Entité/protocole	Câble UTP	Routeur ADSL	802.3	Connecteur RJ45	TCP	IP	hub	switch
Couche	**1**							
Nom de la couche	**Physique**							

Compléter le document Réponse

Le réseau possède une adresse IP de classe C : 192.168.17.0

Question F.1.2
Proposer un plan d'adressage possible pour les différentes machines connectées.

	Routeur ADSL	PC de gestion	PC de commande	Système temps réel
Adresse IP				
Masque				

Compléter le document Réponse

F.2 Maintenance de réseau

Pour des questions de maintenance, le technicien en charge du réseau souhaite effectuer une capture des trames émises par le système temps réel. Pour ce faire il dispose d'un ordinateur portable équipé d'un port 100BaseT / RJ45, et d'un logiciel de capture et d'analyse de trames.

On connecte cet ordinateur sur le commutateur (switch - voir diagramme de déploiement et extrait de documentation en Annexe 8)

Ce commutateur est équipé de 16 ports répartis comme suit :

- Port 1 : noté « UpLink »
- Port 2 et 3 : noté « Replication »
- Port 4 à 15 : ports standards 100 Mbits/sec
- Port 16 : port Gigabit

Le système temps réel est connecté sur le port 4 et le superviseur sur le port 5.

Question F.2.1
Sur quel port faut-il connecter l'ordinateur portable ?

Question F.2.2
La question F.2.1 a-t-elle une raison d'être si l'organe de liaison est un concentrateur et non un commutateur ? Justifier la réponse.

Question F.2.3
Le câble UTP/RJ45 à utiliser est-il un câble croisé ou un câble droit ?

F.3 Client-Serveur

La communication des informations capteur entre le système temps réel et le système de supervision utilise le réseau Ethernet 100baseT et le protocole IP .

On décide d'établir une communication en mode **connecté**. Le système temps réel joue le rôle de serveur ; le PC de supervision est un client de ce dernier. Le port d'écoute fixé par le service est le port **2467**.

Question F.3.1
Les lignes suivantes proposent des solutions en langage C pour l'ouverture de la socket de communication sur le PC de supervision et sur le système temps réel. Pour chaque ligne, préciser, sans justifier, si elle répond aux besoins de l'application.

Solution 1 : `int Sock = socket(AF_INET, SOCK_STREAM, 0) ;`
Solution 2 : `int Sock = socket(AF_INET, SOCK_DGRAM, 0) ;`
Solution 3 : `int Sock = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP) ;`
Solution 4 : `int Sock = socket(AF_UNIX, SOCK_STREAM, 0) ;`
Solution 5 : `int Sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) ;`
Solution 6 : `int Sock = socket(AF_IPX, SOCK_STREAM, 0) ;`
Solution 7 : `int Sock = socket(AF_INET, SOCK_STREAM, IPPROTO_UDP) ;`

La socket de communication étant ouverte avec succès tant au niveau du PC de supervision que du système temps réel, l'application serveur attache maintenant cette socket à un point de communication. Cet attachement (binding) nécessite la fourniture d'une structure de type **sockaddr_in**.

```
struct sockaddr_in
{
    sa_family_t         sin_family; /* address family: AF_INET */
    u_int16_t           sin_port;   /* port in network byte order */
    struct in_addr      sin_addr;   /* internet address */
};

/* Internet address. */
struct in_addr
{
    u_int32_t     s_addr;     /* IPv4 address in network byte order */
};
```

Les questions qui suivent permettent de définir les valeurs des différents membres de la structure de type **sockaddr_in** à fournir à l'appel bind() lors de l'initialisation de la socket serveur.
(prototype : int bind(int Socket, struct sock addr *Name, int NameLen);)

Question F.3.2
Le type **u_int16_t** correspond à un entier non signé sur 16 bits.
Quelle valeur doit être fournie pour le champ **sin_port** de la structure sur le système temps réel?

Question F.3.3
A quoi correspond le champ **in_addr** de cette structure dans ce cas ? Ce champ est souvent initialisé avec la valeur symbolique **INADDR_ANY**. Que signifie cette valeur ?

Question F.3.4
L'appel à **bind()** sera suivi d'un appel à la primitive **listen()** puis à la primitive **accept()**. Quels rôles jouent chacun de ces appels systèmes dans une communication en mode connecté ?

F.4 Dialogue Système Temps Réel / Supervision

Les deux trames reproduites ci-dessous sont extraites d'un relevé réalisé par l'analyseur de protocole lors de l'envoi de la séquence informant la supervision que le robot R1 est passé au-dessus de l'un des marqueurs d'un bain.

L'analyse présente, pour les trames 13 et 14, la description des différents protocoles utilisés : Ethernet, IP, TCP.
Pour chaque protocole, la première ligne donne les caractéristiques principales, les lignes suivantes les valeurs et la signification des différents champs.
La dernière partie donne pour chaque trame le contenu brut en hexadécimal suivi d'une représentation ASCII.

```
Trame   Heure       Adr MAC src     Adr MAC dst   Protocole   Description

13      5.643554    000BDB14E06B    LOCAL         TCP         .AP..., len:    3,

  Frame: Base frame properties
      Frame: Total frame length: 60 bytes
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
    + ETHERNET: Destination address : 00E018B96B0B
    + ETHERNET: Source address : 000BDB14E06B
      ETHERNET: Frame Length : 60 (0x003C)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 46 (0x002E)
  IP: ID = 0x6C12; Proto = TCP; Len: 43
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service

      IP: Total Length = 43 (0x2B)
      IP: Identification = 27666 (0x6C12)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xEB49
      IP: Source Address = 192.168.17.22
      IP: Destination Address = 192.168.17.10
      IP: Data: Number of data bytes remaining = 23 (0x0017)
      IP: Padding: Number of data bytes remaining = 3 (0x0003)
  TCP: .AP..., len:    3, seq:2700201124-2700201127, ack:2053738035, src: 2467  dst: 4425
      TCP: Source Port = 0x09A3
      TCP: Destination Port = 0x1149
      TCP: Sequence Number = 2700201124 (0xA0F1CCA4)
      TCP: Acknowledgement Number = 2053738035 (0x7A698E33)
      TCP: Data Offset = 20 (0x14)
      TCP: Flags = 0x18 : .AP...
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....1... = Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Checksum = 0x3D42
      TCP: Data: Number of data bytes remaining = 3 (0x0003)

00000:  00 E0 18 B9 6B 0B 00 0B DB 14 E0 6B 08 00 45 00   .à.¹k...Û.àk..E.
00010:  00 2B 6C 12 40 00 80 06 EB 49 C0 A8 11 16 C0 A8   .+l.@.€.ëIÀ¨..À¨
00020:  11 0A 09 A3 11 49 A0 F1 CC A4 7A 69 8E 33 50 18   ...£.I ñÌ¤ziŽ3P.
00030:  FA F0 3D 42 00 00 43 06 00 00 00 00               úð=B..C.....

*******************************************************************************


Trame   Heure       Adr MAC src     Adr MAC dst   Protocole   Description

14      5.846679    LOCAL           000BDB14E06B  TCP         .A...., len:    0

  Frame: Base frame properties
      Frame: Total frame length: 54 bytes
  ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
    + ETHERNET: Destination address : 000BDB14E06B
    + ETHERNET: Source address : 00E018B96B0B
      ETHERNET: Frame Length : 54 (0x0036)
      ETHERNET: Ethernet Type : 0x0800 (IP:  DOD Internet Protocol)
      ETHERNET: Ethernet Data: Number of data bytes remaining = 40 (0x0028)
  IP: ID = 0xD73F; Proto = TCP; Len: 40
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
```

```
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 40 (0x28)
      IP: Identification = 55103 (0xD73F)
      IP: Flags Summary = 2 (0x2)
          IP: .......0 = Last fragment in datagram
          IP: ......1. = Cannot fragment datagram
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 64 (0x40)
      IP: Protocol = TCP - Transmission Control
      IP: Checksum = 0xC01F
      IP: Source Address = 192.168.17.10
      IP: Destination Address = 192.168.17.22
      IP: Data: Number of data bytes remaining = 20 (0x0014)
  TCP: .A...., len:    0, seq:2053738035-2053738035, ack:2700201127, src: 4425  dst: 2467
      TCP: Source Port = 0x1149
      TCP: Destination Port = 0x09A3
      TCP: Sequence Number = 2053738035 (0x7A698E33)
      TCP: Acknowledgement Number = 2700201127 (0xA0F1CCA7)
      TCP: Data Offset = 20 (0x14)
      TCP: Flags = 0x10 : .A....
          TCP: ..0..... = No urgent data
          TCP: ...1.... = Acknowledgement field significant
          TCP: ....0... = No Push function
          TCP: .....0.. = No Reset
          TCP: ......0. = No Synchronize
          TCP: .......0 = No Fin
      TCP: Checksum = 0x8053

00000:  00 0B DB 14 E0 6B 00 E0 18 B9 6B 0B 08 00 45 00   ..Û.àk.à.¹k...E.
00010:  00 28 D7 3F 40 00 40 06 C0 1F C0 A8 11 0A C0 A8   .(×?@.@.À.À¨..À¨
00020:  11 16 11 49 09 A3 7A 69 8E 33 A0 F1 CC A7 50 10   ...I.£ziž3 ñÌ§P.
00030:  FA ED 80 53 00 00                                 úí€S..
```

Question F.4.1
Indiquez les adresses Ethernet et les adresses IP du système de supervision et du système temps réel.

	PC de supervision	Système temps réel
Adresse Ethernet		
Adresse IP		

Compléter le document Réponse

Question F.4.2
En identifiant dans la trame 13 le numéro de port source, préciser si cette trame a été émise par le serveur ou par le client.

Question F.4.3
Les paquets IP sont-ils fragmentés ? Justifier la réponse.

Question F.4.4
Les données spécifiques au service de l'application commence à l'octet 36 hexadécimal de la trame 13. Quel est l'identifiant du capteur concerné par cette capture ?

Question F.4.5
Quel est le rôle de la trame 14 ?

Fin du questionnement

BTS INFORMATIQUE ET RESEAUX
POUR L'INDUSTRIE ET LES SERVICES TECHNIQUES

Session 2004

Epreuve E.4
Etude d'un Système Informatisé

Traitement de pièces en ABS par galvanoplastie

Document Réponse (13 pages)

Réponses aux questions B.1

B.1.1 **Nb de transferts =** .....

B.1.2 **Durée théorique minimale :** .....

Durée théorique nominale : .....

B.1.3 **Prétraitement :** .....

Rinçage 1 : .....

Désoxydation : .....

Post-activation .....

Rinçage 2 : .....

Dépôt électrolytique : .....

Rinçage 3 : .....

Réponses aux questions B.2

B.2.1 Nb de transferts = ...

1° transfert : .....

... ... : .....

... ... : .....

... ... : .....

B.2.2 Temps = ...

B.2.3 Distance = ...

B.2.4 Distance = ...

B.2.5 Distance = ...

B.2.6 .....
.....
.....
.....
.....
.....
.....

B.2.7 .....
.....
.....
.....
.....
.....
.....

C.1.1 Opérateur : .....

Opérande : .....

C.1.2 Petite vitesse, sens horaire : ($1001) = ...

Grande vitesse, sens horaire : ($1001) = ...

C.1.3 (pour chaque cas proposé, entourez la bonne réponse)

Module Janz – BigBox : Adéquat – Inadéquat

Motif : .....

.....

Module DIOCOM: Adéquat – Inadéquat

Motif : .....

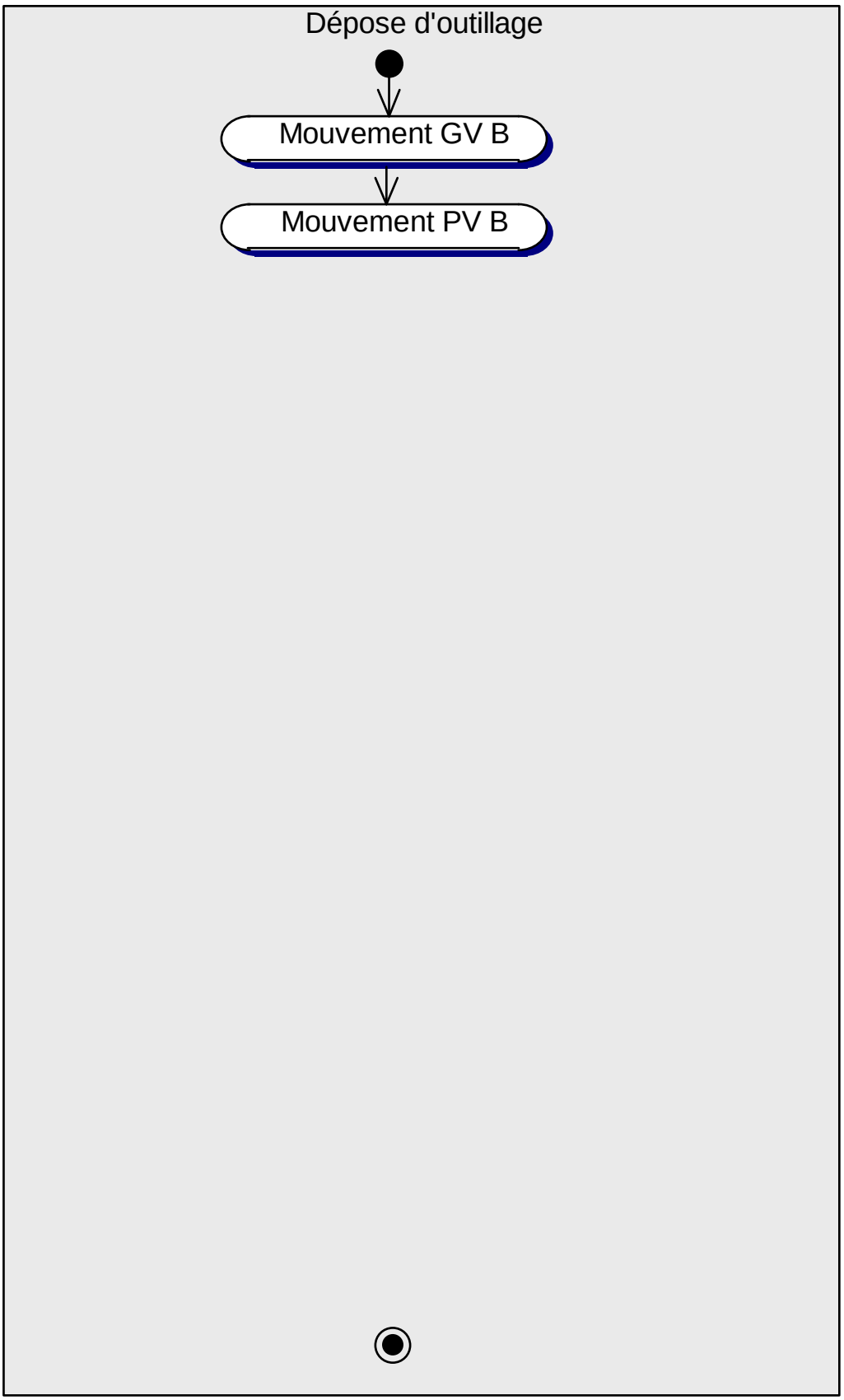
.....

Cartes CIF : Adéquat – Inadéquat

Motif : .....

.....

Réponse à la question C.2



Réponses aux questions C.3

C.3.1 Longueur L = ...

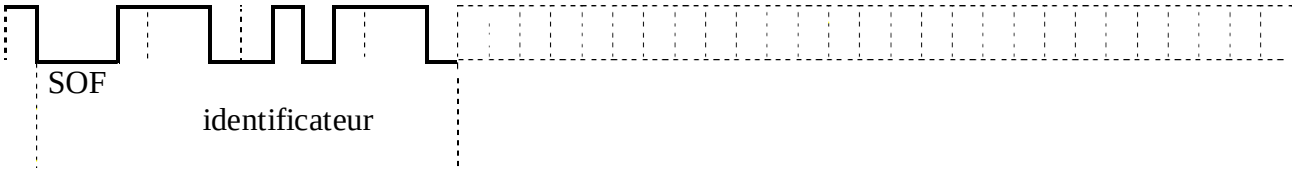
Justification : .....

.....

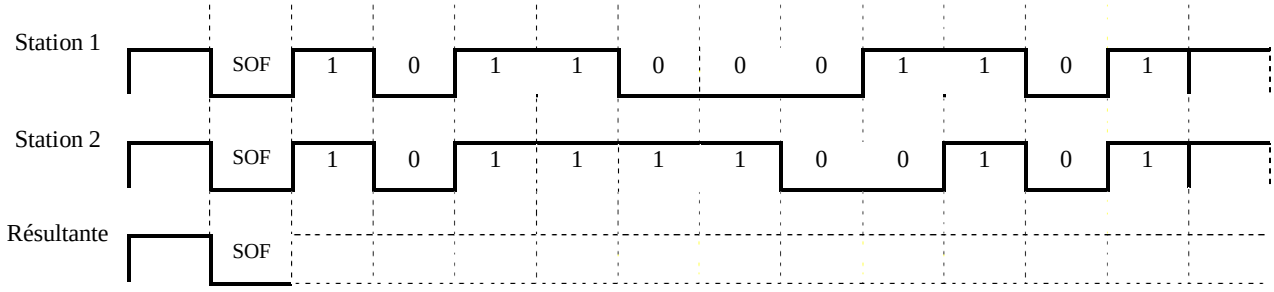
C.3.2 Nb Trames/secondes = ...

C.3.3 Taux de charge = ...

C.3.4



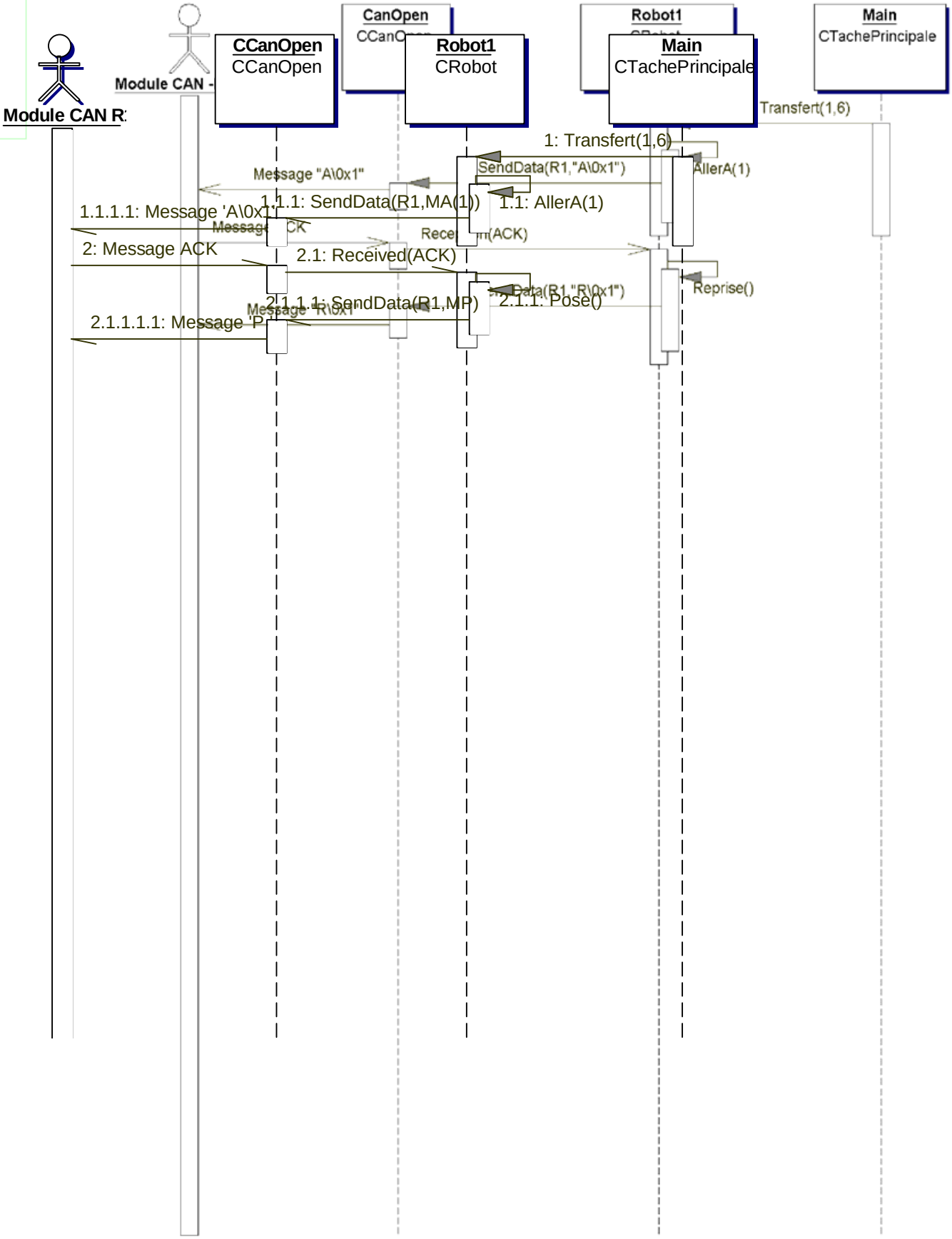
C.3.5



Station émettant effectivement sa trame : **station 1**

station 2

Réponses à la question D.1



Réponses aux questions D.2

D.2.1 .....

.....

.....

D.2.2 **Liste des informations :** .....

.....

.....

.....

.....

.....

D.2.3 .....

.....

.....

.....

.....

.....

D.2.4 .....

.....

.....

.....

.....

.....

Réponses aux questions E.1

E.1.1 classe CRobot :

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

E.1.2 structure de données TOrdre :

.....
.....
.....
.....
.....
.....
.....
.....

E.1.3 structure de données TPosition :

.....
.....
.....
.....
.....
.....
.....

E.1.4 énumération TReponse :

.....
.....
.....
.....
.....
.....
.....
.....
.....

E.1.5 définition de TInfo :

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Réponses aux questions E.2

E.2.1 (justifier si non conforme)

1 : OUI / NON .....
.....

2 : OUI / NON .....
.....

3 : OUI / NON .....
.....

4 : OUI / NON .....
.....

E.2.2 void CRobot::AjoutHisto(TOrdre& ordre)

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

E.2.3 .....

.....
.....
.....
.....
.....

E.2.4 TOrdre CRobot ::DernierOrdre()....................................

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Réponses aux questions F.1

F.1.1

Entité / protocole	Câble UTP	Routeur ADSL	802.3	Connecteur RJ45	TCP	IP	hub	switch
Couche	**1**							
Nom de la couche	**Physique**							

F.1.2

	Routeur ADSL	PC de gestion	PC de commande	Système temps réel
Adresse IP				
Masque				

Réponses aux questions F.2

F.2.1 Port du commutateur = ...

Justification : .....
.....

F.2.2 Oui / Non (entourer la réponse exacte)

Justification : .....
.....
.....

F.2.3 Croisé / Droit (entourer la réponse exacte)

Justification : .....
.....
.....

Réponses aux questions F.3

F.3.1 (Barrer les réponses inexactes)

```
int Sock = socket(AF_INET, SOCK_STREAM, 0) ;
int Sock = socket(AF_INET, SOCK_DGRAM, 0) ;
int Sock = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP) ;
int Sock = socket(AF_UNIX, SOCK_STREAM, 0) ;
int Sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) ;
int Sock = socket(AF_IPX, SOCK_STREAM, 0) ;
int Sock = socket(AF_INET, SOCK_STREAM, IPPROTO_UDP) ;
```

F.3.2 sin_port = ...

F.3.3 in_addr : .....
.....
INADDR_ANY : .....
.....

F.3.4 listen() : .....
.....

accept() : .....
.....

Réponses aux questions F.4

F.4.1

	PC de supervision	Système temps réel
Adresse Ethernet		
Adresse IP		

F.4.2 Trame 13 :

Port source = ...
Emise par : .....

F.4.3 Paquets fragmentés : Oui / Non (entourer la réponse exacte)
Justification : .....
.....
.....

F.4.4 Capteur concerné = ...

F.4.5 Trame 14 : .....
.....
.....

Fin du Document Réponse.

Annexe 1

Définitions et syntaxe

Désignation des capteurs

axe vertical : 4 capteurs inductifs fixes, 1 drapeau métallique mobile

Z	capteur d'initialisation
H	capteur position haute (mouvement horizontal possible)
B	capteur position basse (outillage immergé dans le bain)
E	capteur d'évitement (accrochage / décrochage d'un outillage)

axe horizontal : 1 capteur inductif mobile, N drapeaux métalliques répartis sur la chaîne (suivant nb. de bains)

X	peigne de prise d'origine (à gauche de la zone desservie par le robot)
G	pour un bain donné, capteur à gauche de la position d'arrêt
C	pour un bain donné, capteur d'arrêt à l'aplomb du bain
D	pour un bain donné, capteur à droite de la position d'arrêt

Lorsqu'un module CAN émet une information de type capteur sur le bus CAN, il transmet toujours 16 bits. La lettre clé du capteur (code ASCII) suivie de la valeur numérique sur 8 bits du bain au dessus duquel se trouve le robot.

Désignation des ordres de mouvements

I	prise d'origine machine (initialisation)
S	arrêt inconditionnel (stop)
A(n)	déplacement horizontal à l'aplomb du bain n (aller à)
P	pose d'un outillage (l'outillage est immergé dans le bain)
R	reprise d'un outillage (l'outillage est retiré du bain)

Lorsque le système temps réel émet un ordre en direction d'un module CAN, il transmet toujours 16 bits. La lettre clé du mouvement (code ASCII) suivie de la valeur numérique sur 8 bits du bain de destination pour l'ordre A ; du bain au dessus duquel est censé se trouver le robot pour les ordres P et R ; de la valeur 0 pour les ordres I et S.

remarque : On obtient un jeu de lettres discriminant = A B C D E G H I P R S X Z, lettres utilisées comme codes d'opération dans les trames de dialogue

Désignation des vitesses de mouvement

PV	Petite Vitesse
GV	Grande Vitesse

remarque : le choix de vitesse est géré localement par l'intelligence de pilotage du robot sur le module CAN.

Tâches reconnues par un robot (via son module CAN)

Init()	prise d'origine machine débutant par un déplacement vertical vers le haut (détection du capteur Z) suivi d'un déplacement horizontal vers la gauche jusqu'à détection du peigne de signature d'OM (signal X), puis retour à l'aplomb du poste de chargement (signaux H et C_0).
Stop()	arrêt inconditionnel et instantané de tous les mouvements.
AllerA(n)	déplacement horizontal vers le bain n, signal Cn (valide seulement si position haute).
Transfert(n,m)	transfert de l'outillage du bain n vers le bain m. Ce mouvement complexe se décompose en : AllerA(n) Reprise() AllerA(m) Pose()

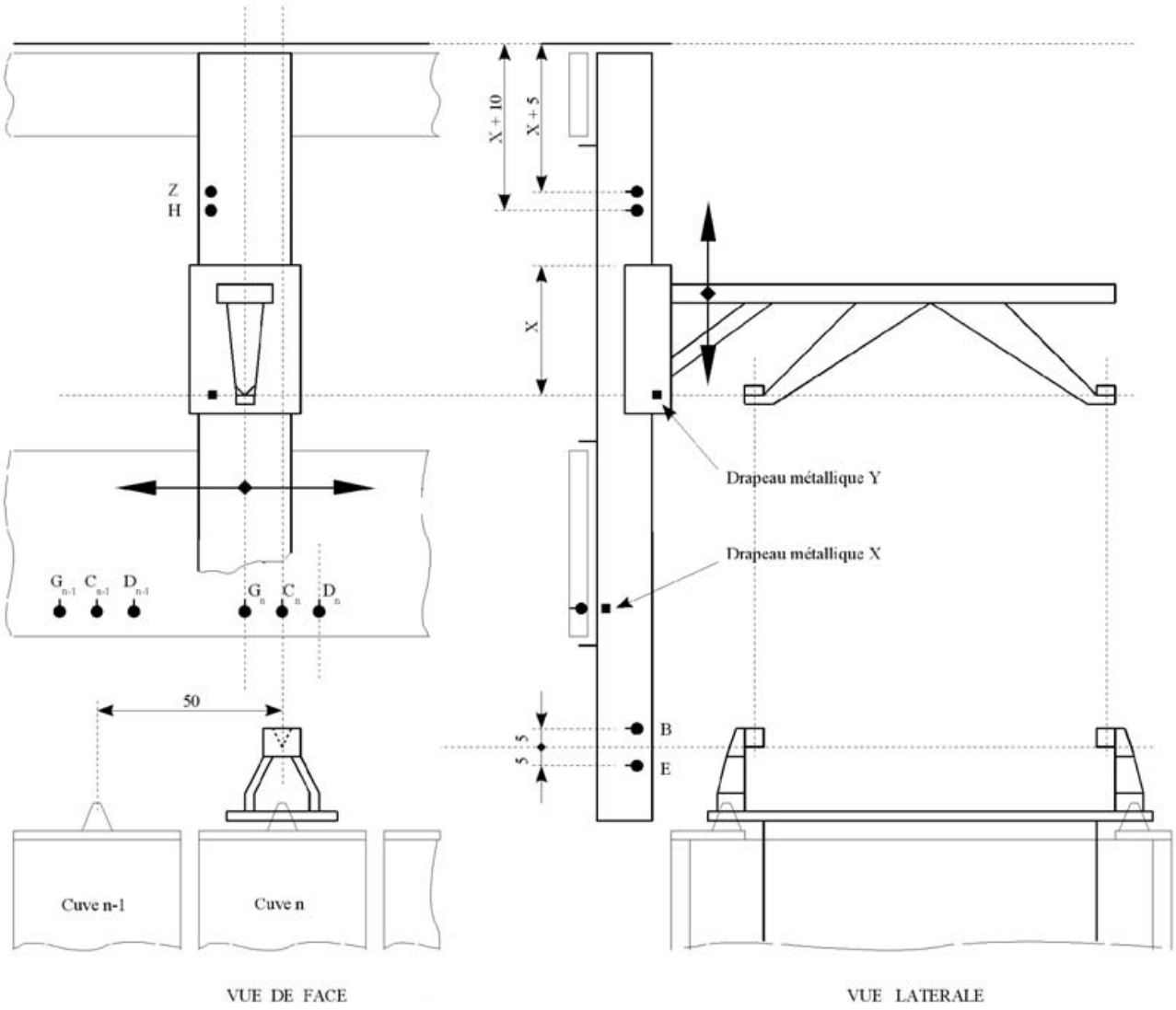
<u>remarque</u> : à l'issue d'exécution des mouvements Init(), AllerA(), Pose() et Reprise(), le robot est toujours arrêté légèrement à droite du capteur C (axe horizontal), et légèrement au-dessus du capteur H (axe vertical).

réponses émises par un robot (via son module CAN)

POM	robot en position d'origine, suite à une commande Init()
ACK	acquittement positif, commande correctement effectuée (sauf pour une demande Init())
NAK	acquittement négatif, échec lors de l'exécution d'une commande

Annexe 2

Capteurs et cycles élémentaires



Le dessin ci-dessus représente le robot en phase de descente vers un outillage.

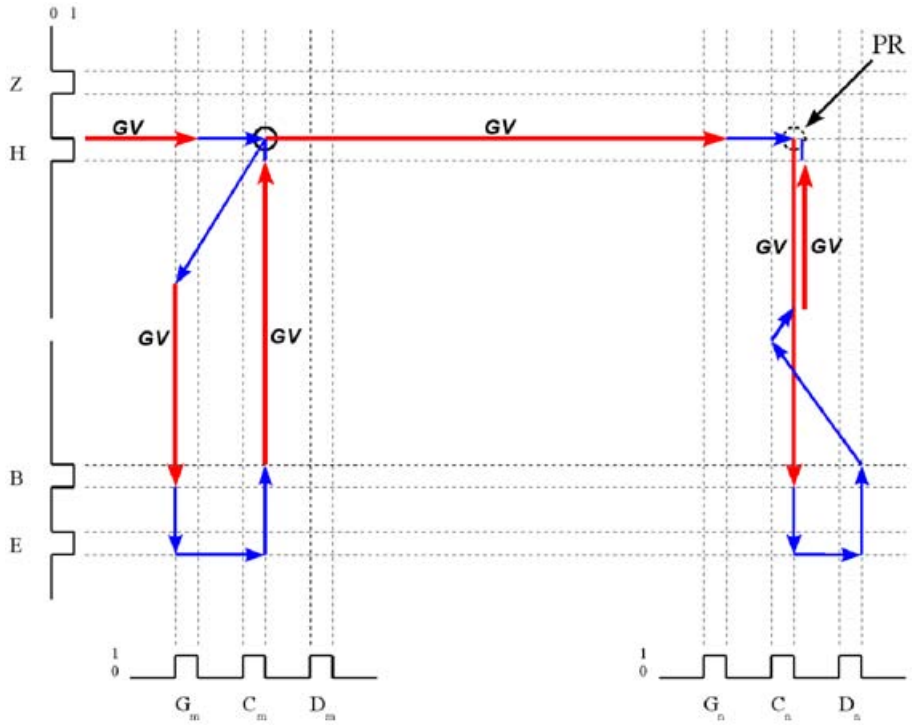
Sur l'axe vertical, les capteurs inductifs sont fixés sur le bâti du robot, le drapeau métallique se déplaçant avec le robot. La vitesse maximale sur cet axe est de 0,2 m/s avec une vitesse faible de 0,05 m/s.

Un premier capteur (Z) est fixé à 5 cm du haut du bâti et n'est utilisé que lors d'une P.O.M. (Prise Origine Machine).

Un second capteur (H) est fixé à 10 centimètres du haut du bâti.

Deux autres capteurs sont fixés en bas du bâti. Le premier (B) situé 5 centimètres au-dessus du plan de dépose des outillages, tandis que le second (E) situé 5 centimètres au-dessous de ce plan et assure la possibilité de dégager horizontalement le robot de l'outillage.

A l'issue de l'exécution d'un mouvement, le robot est toujours arrêté légèrement à droite du capteur C_n (axe horizontal), et légèrement au-dessus du capteur H (axe vertical).



Représentation d'un cycle de transfert Tm,n :

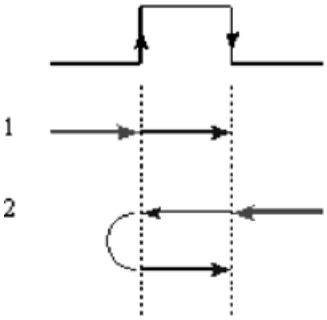
- approche
- prise d'outillage en m
- transfert
- dépose de l'outillage en n

PV = Petite vitesse (défaut).
GV = Grande vitesse

Position de repos (**PR**) :
- à droite de C
- au-dessus de H

Note : Les trois mouvements obliques sont des mouvements combinés ; petite vitesse (PV) sur l'axe horizontal et grande vitesse (GV) sur l'axe vertical.

Les informations capteurs sont représentées en logique positive



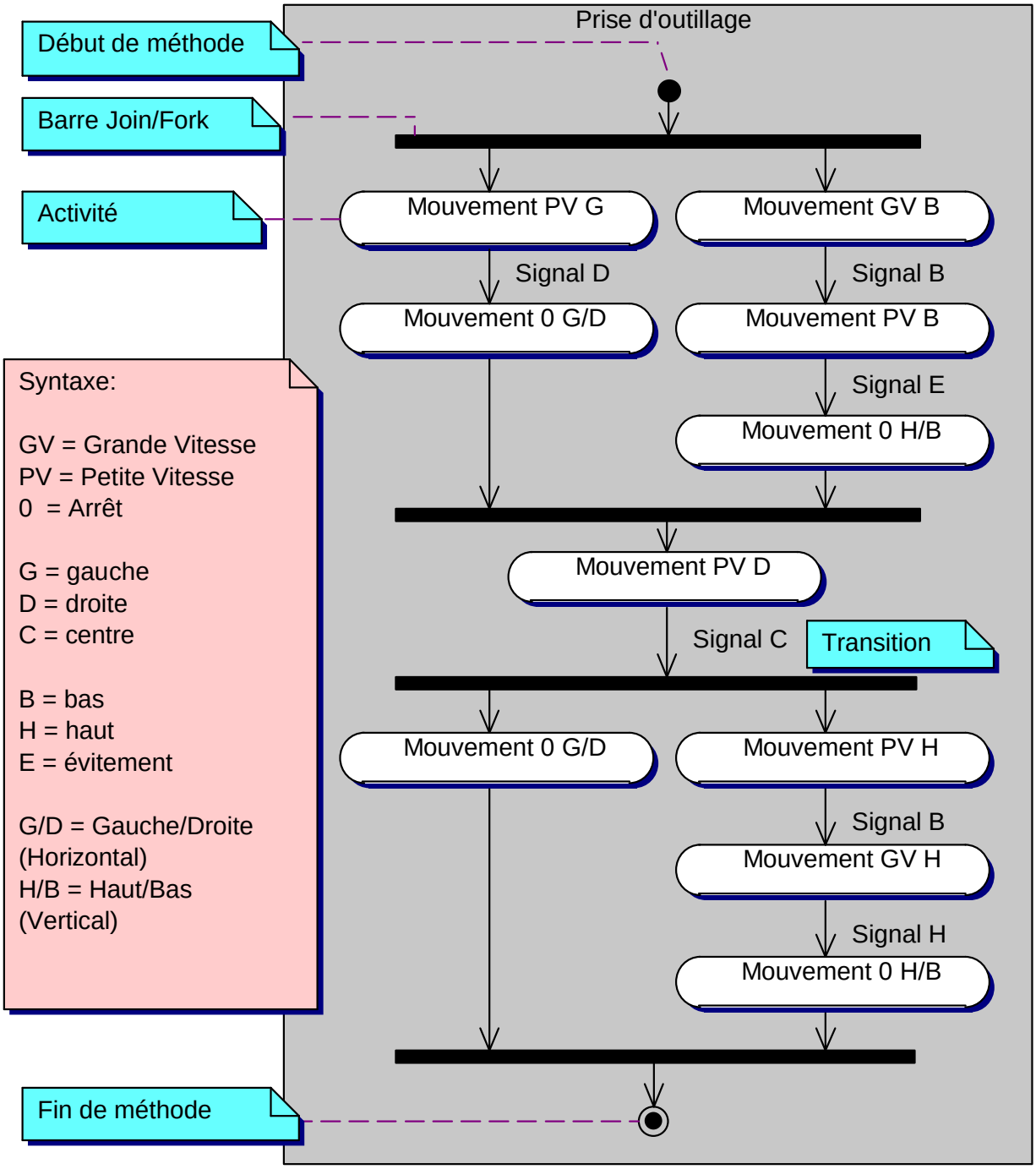
Le système est basé sur la détection des fronts montant et descendant lors du franchissement d'un capteur. Lorsque le robot doit effectivement s'arrêter sur le capteur, la détection du front montant provoque le cas échéant une réduction en PV, jusqu'à obtention du front descendant du même capteur.

La logique diffère suivant le sens d'arrivée sur le capteur... Ci-contre, cas d'un arrêt programmé sur l'axe horizontal :

1 – arrivée par la gauche
2 – arrivée par la droite

Annexe 3

Diagramme d'activité – Prise d'un outillage



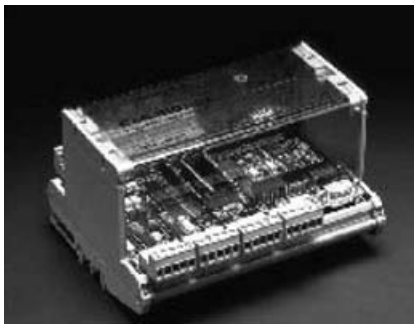
Annexe 4

Equipements – Documents constructeurs

Les données suivantes sont issues de documentations constructeurs.

1. L'équipement CAN-BIGBOX fabriqué par la société JANZ, est un module intelligent permettant de réaliser un système autonome d'acquisition de données connecté sur un bus CAN. Ses principales caractéristiques sont les suivantes :

- micro-controleur MC 68332,
- contrôleur CAN SJA 1000
- 128 à 512 Koctets de Flash EPROM,
- 1 MB de SRAM,
- débit sur bus CAN : 1 Mbit/s,
- température de fonctionnement : 0 à 50 °C
- 8 sorties TOR 24volts,
- 8 entrées TOR 24volts.



2. La famille d'équipement **DIOCOM** de la société LUTZE permet de réaliser des systèmes autonomes d'acquisition de données connectés sur un réseau de terrain ProfibusDP. Leurs principales caractéristiques sont les suivantes :

Caractéristiques communes :
- débit sur le réseau Profibus 1,5 Mbit/s,
- température de fonctionnement : 0 à 60 °C

- Module DC-LB-DI-8-24V : 8 entrées TOR 24 volts,
- Module DC-LB-DI-16-24V : 16 entrées TOR 24 volts,
- Module DC-LB-DO-8-24V : 8 sorties TOR 24 volts,
- Module DC-LB-DO-16-24V : 16 sorties TOR 24 volts,

3. La famille des cartes CIF de la société HILSCHER sont des coupleurs CAN au format des principaux bus PC (ISA, PCI, PCMCIA …).Elles permettent de réaliser des stations de contrôle/commande, des passerelles vers d'autres réseaux etc …

Annexe 5

Documentation bus CAN

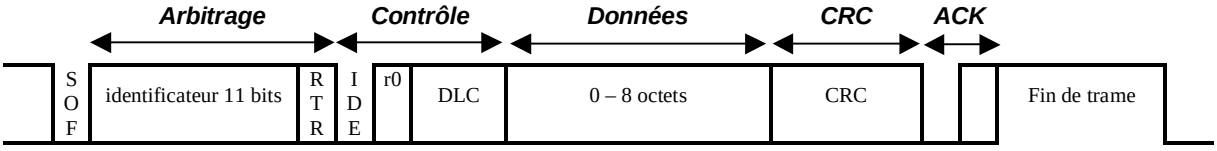
Le bus CAN est un réseau de terrain de type multi-maître dont le débit maximum est de 1 Mbit/s. Le procédé d'attribution du bus est basé sur le principe de l'arbitrage bit à bit, selon lequel les nœuds (ou stations) en compétition, émettant simultanément sur le bus, comparent bit à bit l'identificateur de leur message avec celui des messages concurrents : les stations sont câblées sur le bus par le principe du « ET câblé », et en cas de conflit, c'est à dire émission simultanée, la valeur 0 écrase la valeur 1.

L'état logique **0** est appelé état **dominant**, l'état logique **1**, l'état **récessif**.

Dès qu'une station émettrice se trouvant en état récessif détecte un état dominant, elle s'arrête d'émettre. Tous les perdants deviennent automatiquement des récepteurs du message et ne tentent à nouveau d'émettre que lorsque le bus se libère.

FORMAT DES TRAMES :

La norme CAN définit deux formats de protocole : Standard (version 2.0 A) et Etendue (version 2.0 B). La différence réside dans la longueur de l'identificateur (11 bits en format standard, 29 bits en format étendu).



TRAME AU FORMAT STANDARD

Détail des champs du format standard :

- bit SOF (Start Of Frame) : bit dominant indiquant le début d'une trame,
- champ d'arbitrage : comprend l'identificateur (11 bits) et le bit RTR qui est dominant pour une trame de données, récessif pour une trame de requête. Les bits sont transmis dans l'ordre ID_10 à ID_0 (le moins significatif est ID_0). L'identificateur n'indique pas la destination du message, mais la signification des données du message. Tous les nœuds reçoivent le message et chacun est capable de savoir, grâce à un système de filtrage de messages, si ce dernier lui est destiné ou non,
- champ de contrôle : comprend 6 bits : le bit IDE qui établit la distinction entre format standard (état dominant) et étendu (état récessif), le bit r0 (dominant) réservé pour une utilisation future, et les quatre bits DLC qui indiquent le nombre d'octets contenus dans le champ de données ou le nombre d'octets dont a besoin un nœud du réseau lors d'une trame de requête :

Taille des données en octets	DLC3	DLC2	DLC1	DLC0
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D	D	D

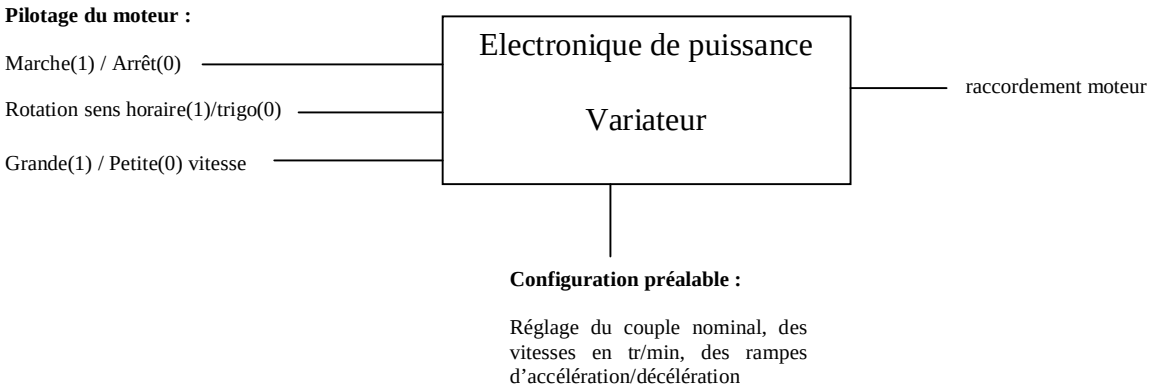
D : Dominant R : Récessif

- champ de données de longueur comprise entre 0 et 8 octets,
- champ CRC de 16 bits (15 bits de CRC suivis d'un bit délimiteur de CRC à l'état récessif),
- champ ACK de deux bits : la station émettrice de la trame laisse le bus libre pendant deux coups d'horloge (ce qui correspond à l'émission de deux bits récessifs) et passe en mode réception pendant le premier coup d'horloge. Le premier bit doit être forcé à l'état dominant par les stations ayant bien reçu la trame. Le second bit est un délimiteur du champ ACK et doit toujours être à l'état récessif
- champ de fin de trame constitué de 7 bits récessifs qui déroge à la règle du « stuffing ».

Règle du « stuffing » : au cours de la construction d'une trame, si 5 bits consécutifs ont la même polarité, un bit de polarité opposée est alors inséré. Ceci évite d'avoir des chaînes de bits identiques trop importantes.

Annexe 6

Pilotage des moteurs des robots.



Annexe 7

Formulaire de mécanique

$$x = \frac{1}{2}\gamma t^2 + V_0 t + x_0 \qquad V = \gamma t + V_0$$

x *est le déplacement en mètres (m).*
x_0 *est la valeur du déplacement à t=0.*

V *est la vitesse en mètres/seconde (m/s).*
V_0 *est la vitesse initiale (à t=0)*

γ *est l'accélération en mètres/seconde² (m/s²)*

t *est le temps en secondes.*

Annexe 8

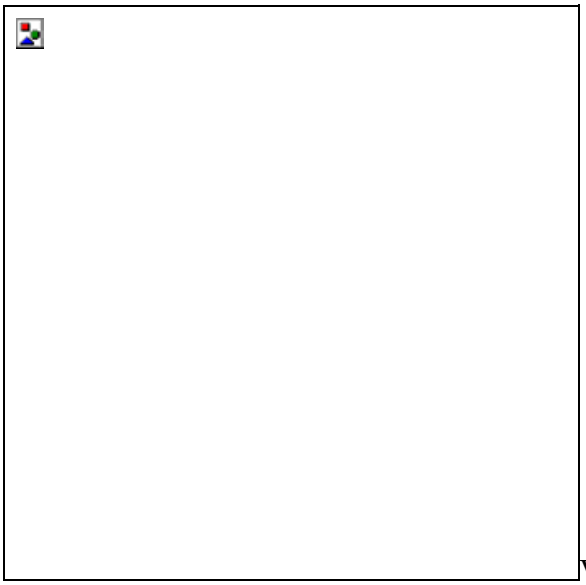
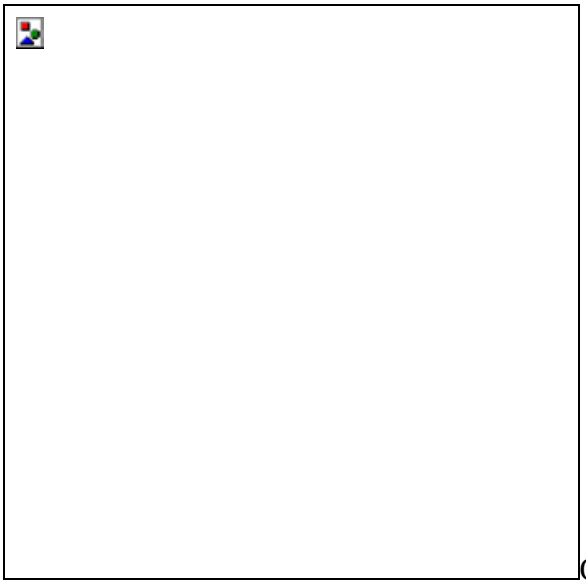
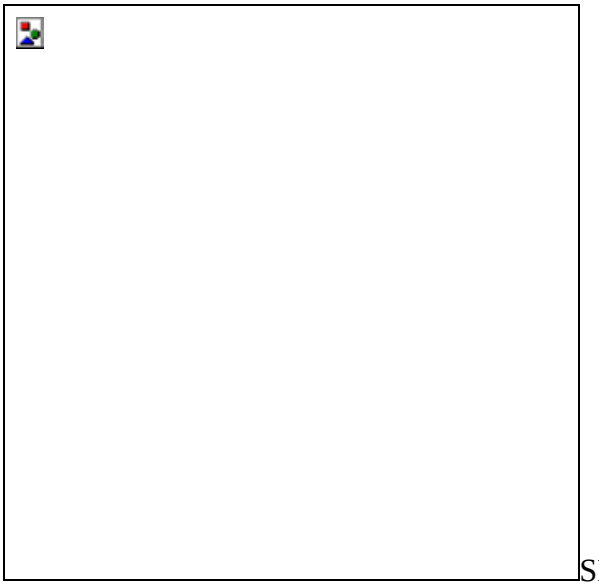
Extrait de documentation – Commutateur DLINK

Commutateur DLINK

Ce commutateur empilable 10/100Mbps de niveau 2 est conçu pour une connexion départementale. Il est muni de 24 ports en standard, et d'un slot d'extension pour recevoir le module d'empilage doté également d'un port GBIC pour une connexion serveur ou dorsale.

Caractéristiques techniques :

- 24 ports :
  - Port 1 : noté « UpLink »
  - Ports 2 et 3 : port « Replication »
  - Ports 4 à 23 : ports standard 100 Mbits/sec
  - Port 24 : port Gigabit

-  Vitesse de fond de panier 8.8Gbps
-  Contrôle de flux 802.3x
-  SNMP, administration Web, monitoring RMON
- Empilable jusqu'à 8 commutateurs
- Fonctions avancées :

- port truncking, VLANs (802.1q) et gestion des priorités,

- GMRP multicast, IGMP Snooping, gestion des priorités 802.1q,
- port mirroring, agrégation des liens.

Note : *La fonctionnalité de " port mirroring " consiste à recopier le trafic d'un ou plusieurs ports sur un autre port (appelé port de Replication) où l'on aura placé un analyseur de protocoles. Cette possibilité permet une analyse de trafic sur un réseau doté d'un commutateur, ce dernier ne renvoyant les paquets reçus uniquement sur le port du destinataire (et non pas sur tous les ports comme un hub ou concentrateur).*