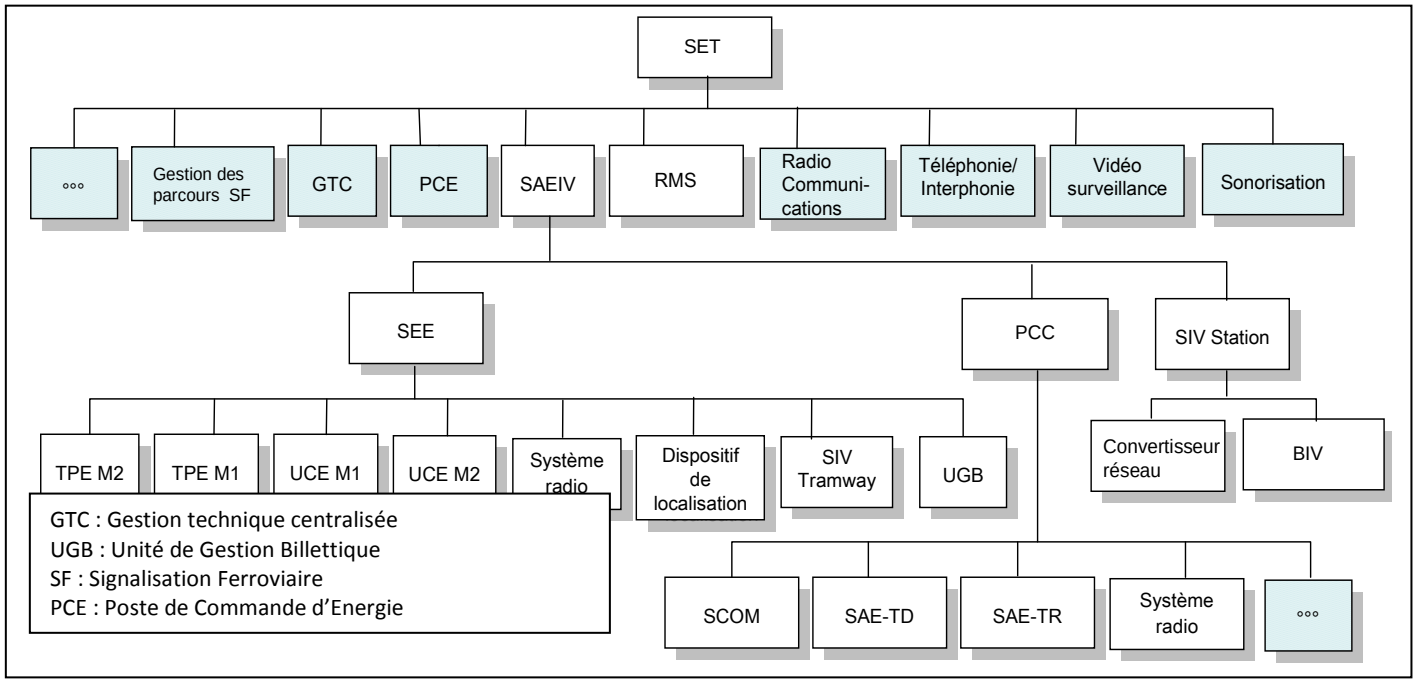
BTS INFORMATIQUE ET RÉSEAUX POUR L'INDUSTRIE ET LES SERVICES TECHNIQUES

Session 2010
ÉPREUVE E.4
Étude d'un Système Informatisé

AIDE À L'EXPLOITATION D'UN TRAMWAY ET INFORMATION DES VOYAGEURS

ANNEXES

Annexe 1 : **Schémas d'architecture du SAEIV**



Principaux sous-systèmes et fonctions du Système d'Exploitation du Tramway (SET)

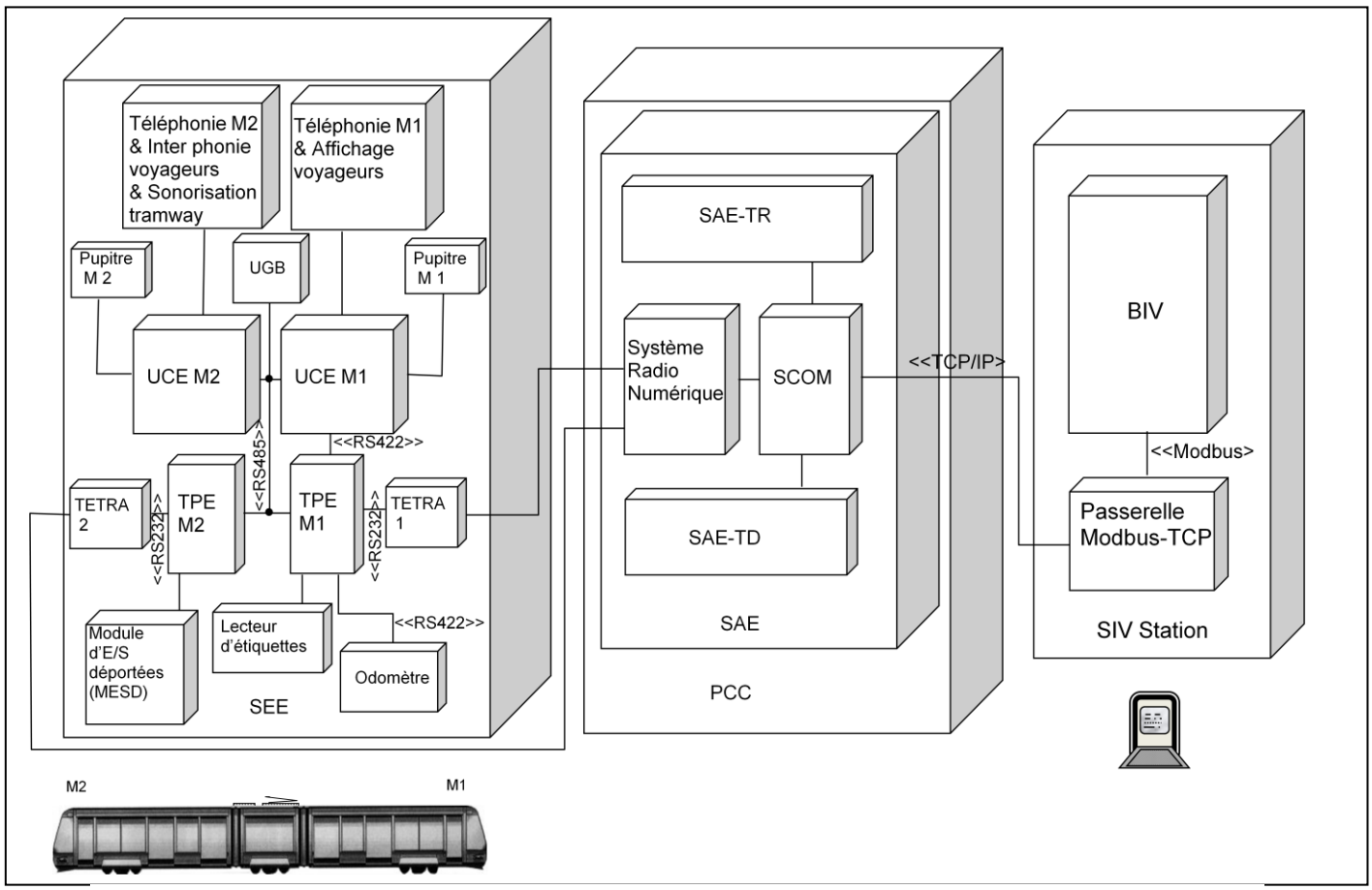
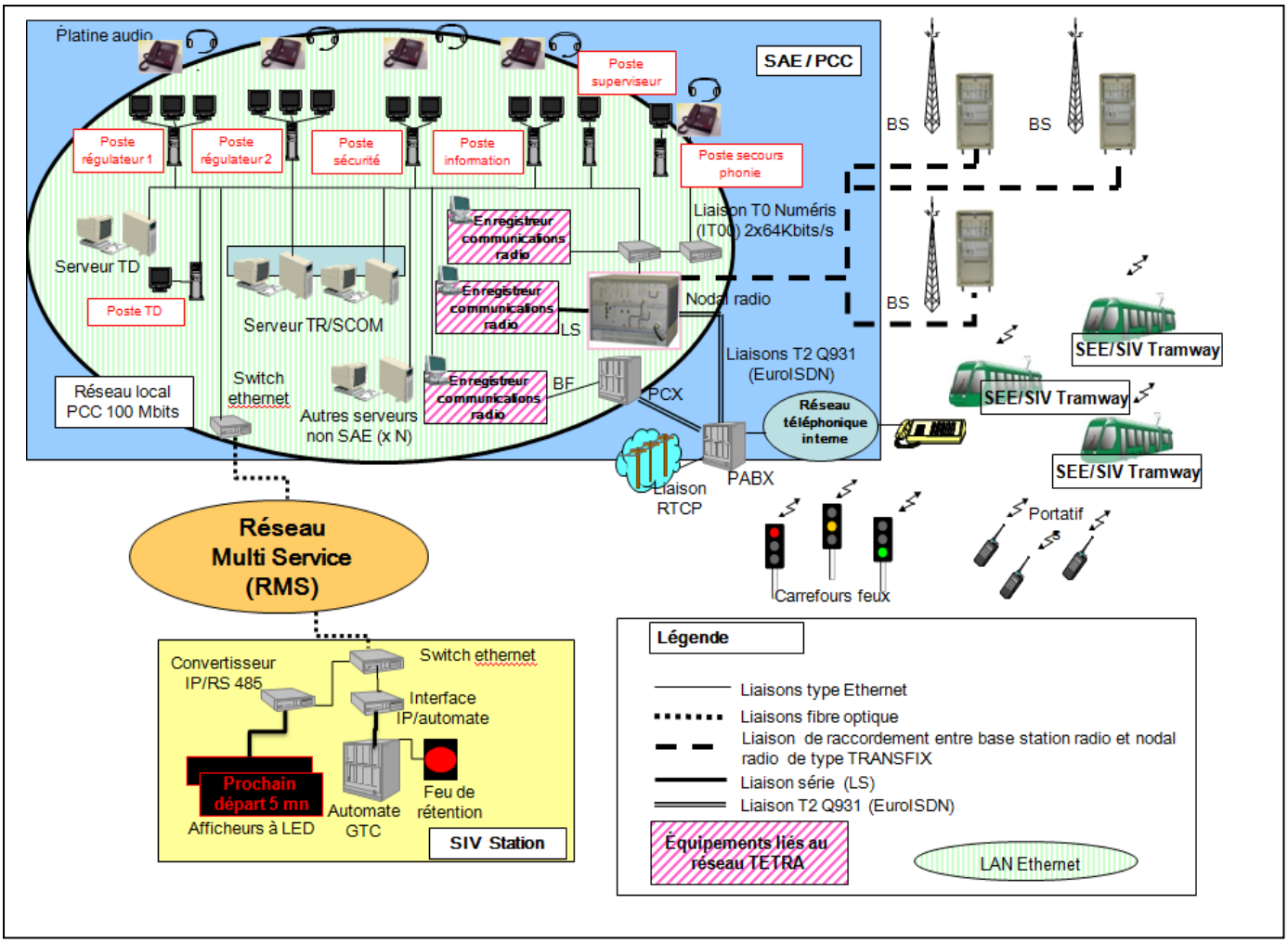
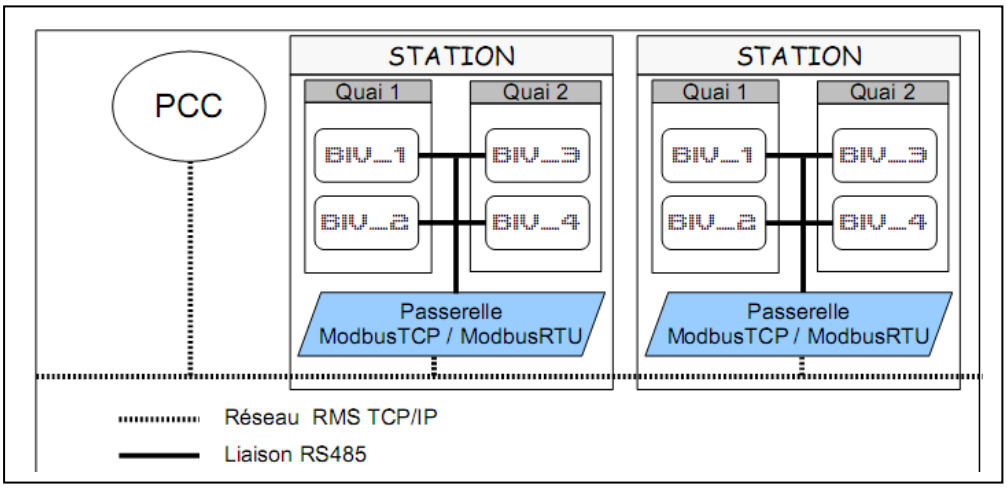


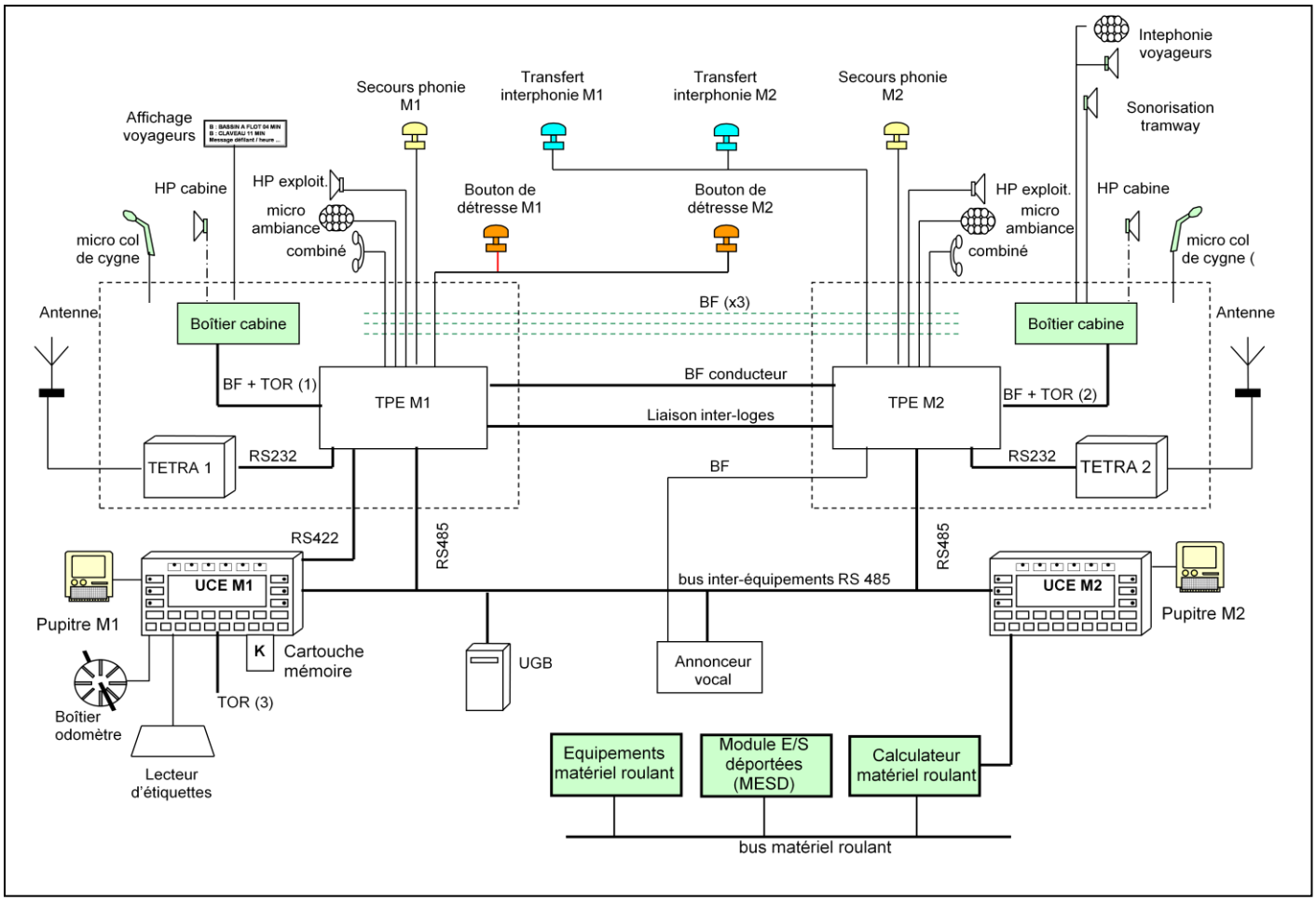
Diagramme des nœuds du SAEIV



Architecture générale du Système d'Aide à l'Exploitation et d'Information des voyageurs (SAEIV)



Architecture générale des Bornes d'Information Voyageurs) BIV du SIV Station



Architecture générale du Système d'Exploitation Embarqué (SEE)

Légende :

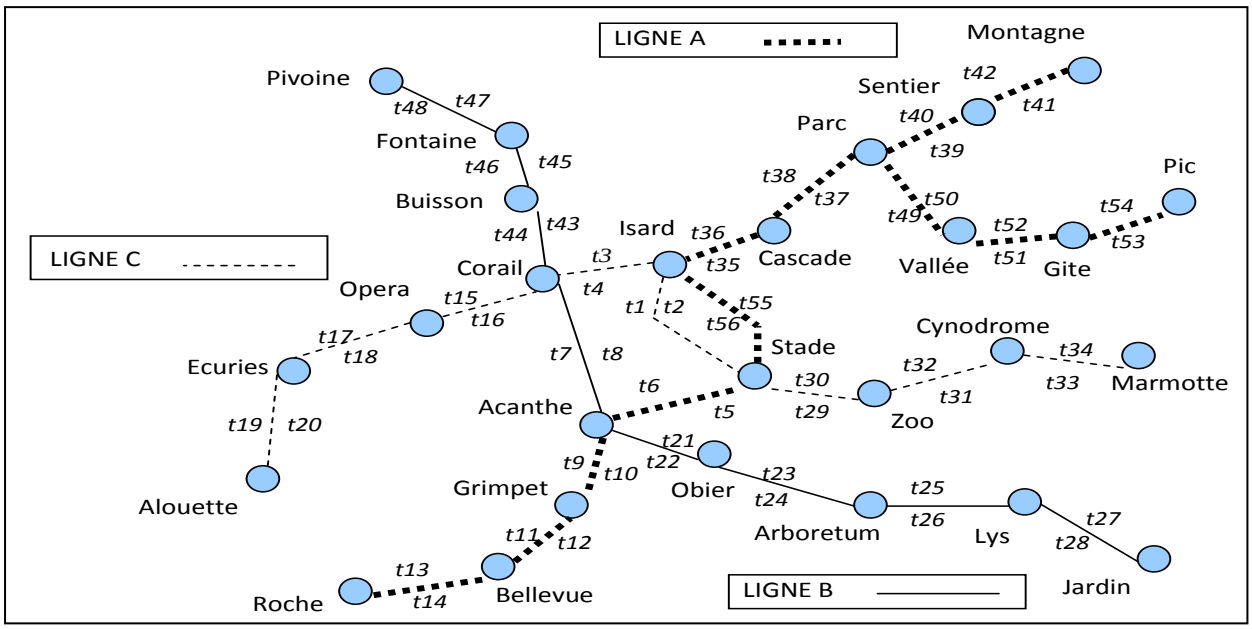
(1) : BF conducteur issue du micro col de cygne (Matériel Roulant → TPE M1)
BF affichage informations voyageurs (TPE M1 → Matériel Roulant)
TOR sonorisation conducteur en cours (Matériel Roulant →TPE M1)
TOR demande interphonie cabine / cabine (Matériel Roulant → TPE M1)
TOR acquittement interphonie voyageurs (Matériel Roulant → TPE M1)
TOR demande main libres (TPE M1 → Matériel Roulant)

(2) : BF conducteur issue du micro col de cygne (Matériel Roulant → TPE M2)
BF sonorisation (TPE M2 → Matériel Roulant)
BF interphonie voyageurs (TPE M2 → Matériel Roulant)
TOR sonorisation conducteur en cours (Matériel Roulant → TPE M2)
TOR demande annonce PCC ou annonce sonore (TPE M2 → Matériel Roulant)
TOR demande transfert interphonie (TPE M2 → Matériel Roulant)
TOR mise en communication (TPE M2 → Matériel Roulant)

(3) : TOR sonorisation conducteur en cours (Matériel Roulant → UCE M1)
BP priorité feux manuelle issue de M1et M2 (Matériel Roulant → UCE M1)
UGB Unité de Gestion Billettique

Annexe 2 : **Topologie et référentiel SAE**

1. Topologie du réseau du tramway (trois lignes)



2. Glossaire du réseau du tramway

LIGNE : Une ligne est définie comme un ensemble de chaînages ; eux mêmes constitués d'un ensemble ordonné de points d'arrêt allant d'un terminus départ à un terminus arrivée. L'arrêt de fin de ligne indique la destination. Une station regroupe des arrêts. Une ligne est décrite par :

- un numéro identifiant la ligne (sur 3 chiffres donc 255 lignes maximum),
- un mnémonique identifiant la ligne (sur 4 caractères),
- un libellé de ligne. Ce libellé doit correspondre obligatoirement à la lettre A, B ou C suivant la ligne car il sert ensuite au SAE-TR pour envoyer des informations aux autres sous-systèmes (TCO, GTC) en cas de changement de mnémonique de ligne.

PARCOURS (chaînage d'arrêts) : Un parcours est un itinéraire destiné à être parcouru dans sa totalité par un véhicule. Il est décrit par la suite ordonnée des arrêts qui le composent. Le premier arrêt de la liste est le terminus départ, le dernier le terminus arrivée.
Un parcours est défini par :

- le sens de parcours du chaînage : **A** pour Aller et **R** pour Retour,
- le numéro d'ordre du chaînage (sur 2 chiffres donc 99 chaînages pour une ligne),
- le nom du chaînage (sur 32 caractères),
- une destination pour l'information des voyageurs,
- une voie.

ETIQUETTE DE RECALAGE : Elles permettent la localisation du tramway en des points précis du réseau. Une étiquette de recalage est définie par :

- un numéro qui identifie l'étiquette (0 à 999),
- une adresse (0 à 255),
- son type.

TRONCON DE VOIE (noté Txx sur la figure « Topologie des 3 lignes de réseau de tramway) :
Un tronçon de voie est défini par :

- un libellé identifiant le tronçon (24 caractères maxi.),
- une longueur (1 à 20000 mètres),
- deux étiquettes de recalage.

.**VOIE :** Une voie est définie par :

- un libellé,
- un chaînage de tronçons de voies orientés. L'orientation du tronçon de voie se détermine suivant qu'il est dans le sens conventionnel (arbitraire, déterminé lors de la création en fonction du positionnement des objets sur le tronçon de voie) ou dans le sens inverse.

3. Référentiel SAE embarqué

3.1. Représentation tabulaire (extrait)

Le référentiel embarqué dans chaque tramway consiste en un certain nombre de tables qui contiennent la description des itinéraires à parcourir.
Cette description est fournie sous la forme d'un ensemble de fichiers (référentiel) décrivant :

- pour un service matériel donné, la liste des courses successives à réaliser et pour chacune d'entre elles l'itinéraire qu'elle emprunte,
- la description de ces itinéraires sous la forme d'une liste ordonnée de points décrivant pour chacun d'entre eux la distance curviligne qui le sépare du point suivant,
- la description de la position des étiquettes sur les voies.

La constitution de ces fichiers est réalisée par le SAE-TD qui les met à la disposition du serveur de communication.
Leur transfert vers les équipements embarqués est assuré par le serveur de communication puis le réseau de radio numérique TETRA.

La liste des parcours à effectuer est susceptible d'être modifiée, notamment aux terminus. Elle est signalée par l'émission par le SAE-TR de messages radio via le réseau radio numérique TETRA. Ces messages contiennent les modifications des fichiers référentiels permettant aux équipements embarqués de continuer à assurer leur localisation.

Les distances des tronçons sont celles de centre à centre de stations consécutives.

LIGNE		
ID_Ligne	Mnémo	Libelle
97	A	A
98	B	B
99	C	C

ARRET			**ARRET** (suite)			**ARRET** (suite)	
ID_Arret	Nom		11	Grimpet		22	Fontaine
1	Montagne		12	Bellevue		23	Buisson
2	Sentier		13	Roche		24	Obier
3	Parc		14	Alouette		25	Arboretum
4	Pic		15	Opera		26	Lys
5	Gite		16	Ecuries		27	Jardin
6	Vallée		17	Corail			
7	Cascade		18	Zoo			
8	Isard		19	Cynodrome			
9	Stade		20	Marmotte			
10	Acanthe		21	Pivoine			

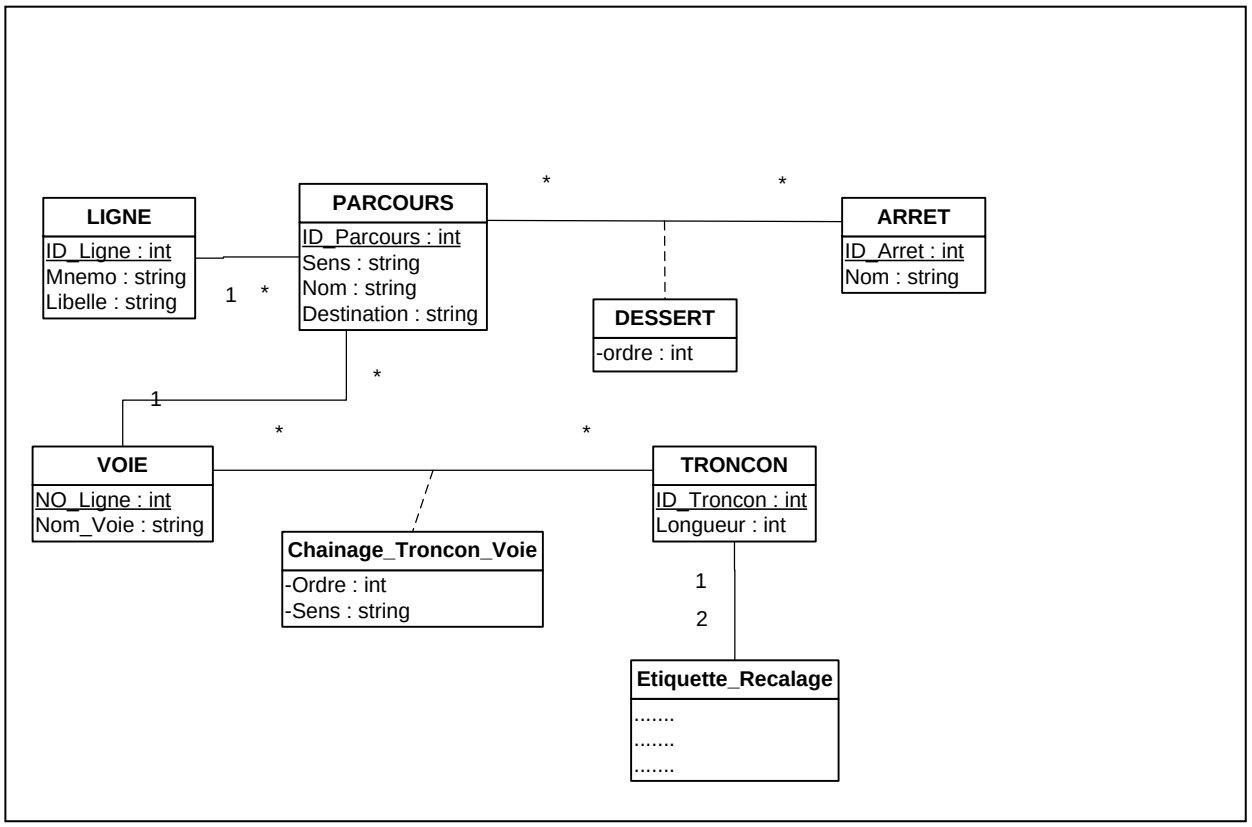
PARCOURS (chainage)					
ID_Parcours	Sens	Nom	Destination	ID_Ligne	Voie
3	Aller	Montagne-Cascade	Roche	97	66
4	Aller	Pic-Isard	Roche	97	76
5	Aller	Isard-Roche	Roche	97	88
10	Aller	Stade-Alouette	Alouette	99	98
11	Aller	Marmotte-Stade	Alouette	99	98
12	Aller	Pivoine-Buisson	Jardin	98	54
6	Aller	Buisson-Acanthe	Jardin	98	54
7	Aller	Acanthe-Jardin	Jardin	98	44
8	Retour	Montagne-Cascade	Montagne	97	67
13	Retour	Pic-Isard	Pic	97	77
14	Retour	Isard-Roche	Montagne	97	89
21	Retour	Stade-Alouette	Marmotte	99	99
22	Retour	Marmotte-Stade	Marmotte	99	99
23	Retour	Pivoine-Buisson	Pivoine	98	55
9	Retour	Buisson-Acanthe	Pivoine	98	55
15	Retour	Acanthe-Jardin	Pivoine	98	45

DESSERT		
ID Parcours	**ID Arret**	Ordre
3	1	1
3	2	2
3	3	3
3	7	4
4	4	1
4	5	2
4	6	3
4	3	4
4	7	5
4	8	6
5	8	1
5	9	2
5	10	3
5	11	4
5	12	5
5	13	6
10	9	1
10	8	2
10	17	3
10	15	4
10	16	5
10	14	6
11	20	1
11	19	2

DESSERT(suite)		
11	18	3
11	9	4
12	21	1
12	22	2
12	23	3
6	23	1
6	17	2
6	10	3
7	10	1
7	24	2
7	25	3
7	26	4
7	27	5
8	7	1
8	3	2
8	2	3
8	1	4
13	8	1
13	7	2
13	3	3
13	6	4
13	5	5
13	4	6
14	13	1
14	12	2

DESSERT(suite)		
14	11	3
14	10	4
14	9	5
14	8	6
21	14	1
21	16	2
21	15	3
21	17	4
21	8	5
21	9	6
22	9	1
22	18	2
22	19	3
22	20	4
23	23	1
23	22	2
23	21	3
9	10	1
9	17	2
9	23	3
15	……	……
15	……	……
15	……	……
15	……	……
15	……	……

3.2. Modèle conceptuel de données en UML (extrait)



Annexe 3 : **Boîtier odomètre**

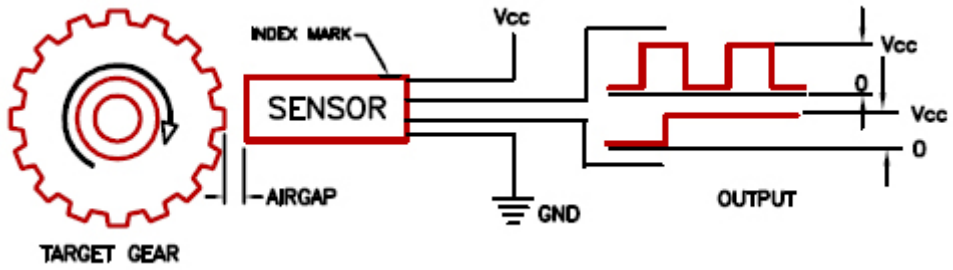
1. Description

Le boîtier odomètre est un boîtier intelligent conçu par le constructeur du tramway. Il comporte une entrée pour le capteur odométrique, un microcontrôleur comportant des temporisateurs, et un connecteur d'interface à la norme V11, c'est-à-dire pour des liaisons séries RS422 et RS485.

L'odomètre dispose d'un capteur magnétique réagissant avec une roue odométrique crénelée attenante à une roue du tramway. Ce capteur est basé sur des détecteurs à effet Hall, et est capable de détecter le sens de rotation de la roue.

A partir des informations fournies par le capteur, le boîtier odomètre élabore plusieurs données :

- position odométrique
- vitesse
- temps de roulage et temps d'immobilisation



Extrait de la documentation du capteur :

Non-contact magnetic sensors that measure the distortion of magnetic fields and thus provide precise measurements of speed and direction. Output #1 is digital square wave and measures the speed of target wheel or gear. Output #2 is a DC level that when the target wheel rotates clockwise, the output signal # 2 produces logic High, and when the target wheel rotates counter clockwise, the output signal # 2 produces logic low. Output signal #1 will be 50% duty cycle with proper alignment of sensor and target gear.

2. Calcul de la position odométrique

La position odométrique est obtenue par un simple comptage ou décomptage des impulsions fournies par le capteur, selon que le sens de rotation indiqué par le capteur est positif ou négatif.

Une commande spécifique envoyée par la liaison série permet de remettre le compteur à zéro. La position odométrique est donc toujours relative à la dernière remise à zéro.

L'odomètre d'un véhicule a besoin d'être étalonné, en indiquant le diamètre des roues du tramway et le nombre de créneaux de la roue odométrique. De là, il est facile de déduire la distance parcourue par le tramway entre 2 impulsions du capteur.

3. Protocole de communication du boîtier odomètre

L'odomètre est commandé par l'envoi de trames d'octets de commande sous la forme suivante :

COMMANDE	VALEUR	Commentaire/exemple
RAZ	STX | Z | ETX	Remet à 0 la position odométrique, le temps de roulage et le temps d'immobilisation.
Etalonnage	STX | E | diamètre ASCII 3 octets | Nb creneaux ASCII 2 octets | ETX	Fournit le diamètre de la roue de tramway (en mm) et le nombre de créneaux de la roue odométrique. Ex, diamètre 600 mm 64 créneaux : <stx>E60064<etx>
Démarrage	STX | D | ETX	Met le boîtier odomètre en état de fonctionnement.
Arrêt	STX | A | ETX	Met le boîtier odomètre à l'état arrêté.
Demande mesure	STX | M | ETX	Demande une trame comportant les diverses mesures effectuées par le boîtier.

Les positions sont exprimées en mètres (m), les vitesses en mètres par seconde (m/s) et les temps en secondes (s).

En réponse aux trames de commandes 'Z', 'E', 'D', 'A', l'odomètre envoie la trame de réponse :

REPONSE	VALEUR	Commentaire/exemple
OK	STX | O | K | ETX	Opération réalisée
NOT OK	STX | N | Numéro erreur ASCII 2 octets | ETX	Problème

En réponse à une commande ‘M’ (Demande mesure), s'il ne détecte pas de problème, l’odomètre envoie la trame de réponse suivante, de longueur fixe. Cette trame contient 4 chaînes de caractères terminées chacune par un NULL, et encadrées par <STX> et <ETX> :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
STX	Vitesse linéaire (ASCII: 6 Octets)						NULL	Position odométrique (ASCII: 7 Octets)							NULL	Temps roulage (ASCII: 5 Octets)					NULL	Temps immobilisation (ASCII: 5 Octets)					NULL	ETX

En cas le problème, l'odomètre renvoie une trame « NOT OK ».

Exemple de trame réponse à la commande M (les caractères non imprimables sont exprimés en hexadécimal) :

0x02	**0**	**0**	**4**	**6**	**.**	**4**	**0x00**	**-**	**0**	**4**	**0**	**2**	**.**	**8**	**0x00**	**0**	**0**	**6**	**4**	**8**	**0x00**	**0**	**3**	**5**	**9**	**1**	**0x00**	**0x03**

Annexe 4 : **Lecteur d'étiquettes Balogh**

1. Rôle du lecteur d'étiquettes

Les informations de localisation absolue sont fournies par un dispositif constitué :
- d'un lecteur d'étiquettes installé sous la rame,
- d'étiquettes fixées sur la voie.

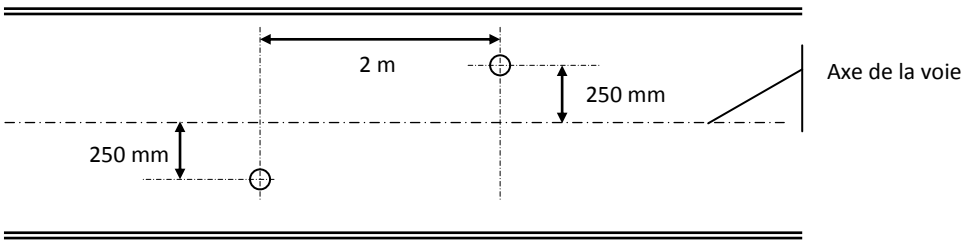
Lors du passage de la rame au dessus d'une étiquette, un échange s'établit entre cette étiquette et le lecteur, permettant à celui-ci d'acquérir l'identification de l'étiquette et de la transmettre à l'UCE.

Ces informations permettent le recalage de la position absolue du véhicule.

Le lecteur d'étiquettes assure :
- le signalement de la détection d'une étiquette,
- l'instant de lecture de l'étiquette,
- le numéro d'étiquette (identifiant physique de l'étiquette),
- la fourniture de son état de fonctionnement.

2. Positionnement des étiquettes

Les étiquettes sont implantées en couple en entre-rails, selon la disposition suivante :



Un couple d'étiquettes de re-localisation est systématiquement placé :
- en entrée de station voyageur sur chaque voie, à 10m du centre de la station. Ce sont les étiquettes « amont ».
- en sortie de station voyageur sur chaque voie, à 10m du centre de la station. Ce sont les étiquettes « aval ».
- aux points d'entrée-sortie d'un dépôt.
- en sortie de chaque aiguillage.

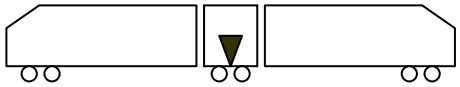
Les étiquettes réceptrices implantées sur le réseau tramway sont des étiquettes Balogh. Il s'agit d'étiquettes passives réagissant sur émission du lecteur fixé sur le boggie de la rame.

Les étiquettes d'un même couple sont programmées avec la même identité.

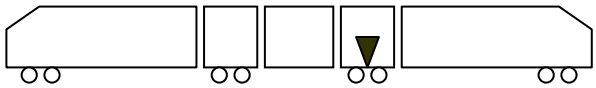
3. Positionnement des lecteurs

Un lecteur d'étiquettes Balogh est connecté à l'Unité Centrale Embarquée (UCE) M1.
Il permet l'acquisition des étiquettes de voie franchies par le tramway.
Le dialogue avec le lecteur est assuré au travers d'une liaison série RS422.
L'antenne de lecture des étiquettes est logée sous le bogie porteur.

Sur une rame courte, le bogie porteur est au centre de la rame :



Sur une rame longue, le bogie porteur est accolé à l'élément M2 :



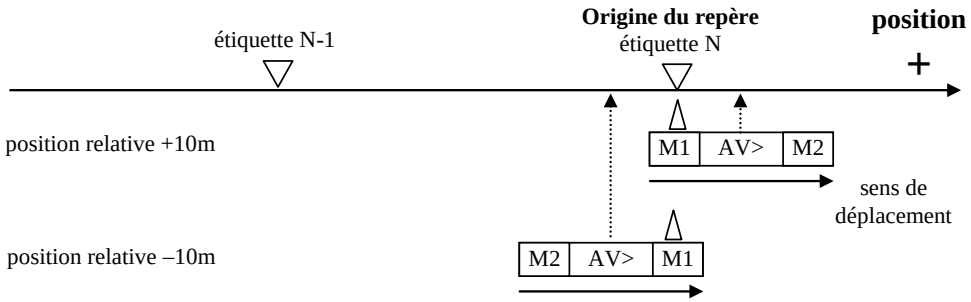
Longueur de la rame: courte Longueur de la rame: longue	32 m 42 m
Position du lecteur d'étiquettes / M1: rame courte Position du lecteur d'étiquettes / M1 rame longue	16 m 26 m

4. Position absolue par étiquettes

La position absolue par étiquettes est une donnée identifiant la position du véhicule.
C'est une donnée structurée comportant :

- le numéro de la dernière étiquette lue (étiquette N)
- le numéro de l'avant-dernière étiquette lue (étiquette N-1)
- la position relative à la dernière étiquette, comptée positivement dans le sens N-1 -> N .

La position du véhicule est toujours exprimée pour le **centre de la rame**.
Si le lecteur d'étiquettes n'est pas au centre de la rame, la position du véhicule est corrigée en fonction de la longueur de la rame et de la distance du lecteur à la tête de rame (loge M1).



5. Recalage et relocalisation sur étiquette

La lecture d'une étiquette, lorsqu'elle est attendue sur l'itinéraire du parcours effectué, déclenche un recalage automatique de la position du véhicule.

Le recalage consiste à :

- calculer la position absolue en fonction de l'étiquette détectée
- remettre à zéro le boîtier odomètre

Le véhicule est déclaré « hors ligne » suite à la lecture d'une étiquette non attendue sur le parcours effectué ou si une étiquette n'a pas été lue depuis plus de 10 mn (panne possible du tramway).

Suite au passage « hors ligne » le véhicule peut être relocalisé suite à la lecture d'une étiquette prévue sur le parcours qu'il est censé effectuer. Cette relocalisation peut également être réalisée manuellement par le conducteur.

6. Localisation périodique des rames de tramway

Chaque rame de tramway calcule périodiquement sa localisation à partir de la distance odométrique fournie par le boîtier odomètre et de la position absolue obtenue lors de la dernière remise à zéro du boîtier.

7. Documentation technique du matériel

189, rue d'Aubervilliers CP 97 75886 PARIS Cedex18 FRANCE
Tél : 33 (0)1.44.65.65.00 Fax : 33 (0)1.44.65.65.10
http://www.balogh-rfid.com

IDENTIFICATION SYSTEMS

Control board

BELS 160L/STD

DESCRIPTION

The BELS 160L/STD is an IP65-rated slave interface exchanging data through an RS-422 / RS-485 serial link using a specific protocol.

It features one channel for the assembly ANT 80LP+ERL120 to read the code of an OLR 85 tag and store it.

HOST COMMUNICATION PROTOCOL

Communication with the host can be configured for :
- automatic transmission of the read code,
- polling (uploading upon user request): BELS answers the request <a> by transmitting the last read code or a fault code.

Various frames transmitted:

• read data:	<STX>	<A>	<E>	<0>	<1st byte>	<2nd byte>	<CS>
• board fault report:	<STX>	<A>	<D>	<E>	<F>	95h	<CS>
• notice of tag absence upon power-up:	<STX>	<A>	<N>	<O>	<R>	<M>	<CS>

<CS> stands for the frame checksum XOR-derived from the preceding six bytes.

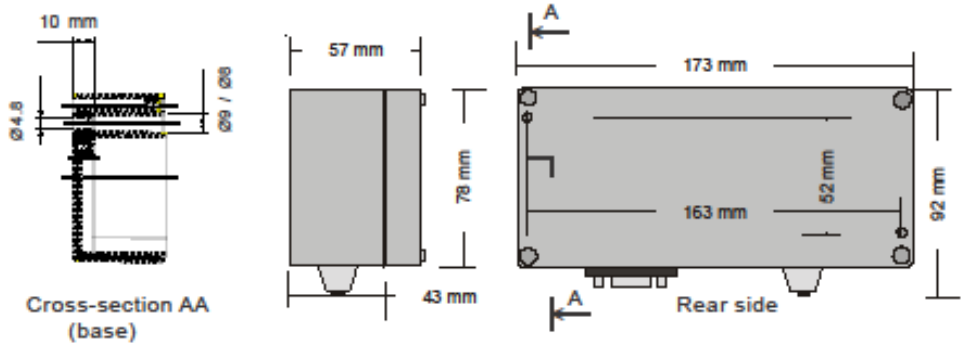
DATA FOR ASSEMBLY

Mounting:

Overhanging using two M4 screws, head Ø 6...7.5 mm, length under head > 20 mm (see cross-section AA).
Right-angled distances: 163 x 52 mm as shown on the hereunder rear side view.

Connection to the ANT80LP+ERL120: the M12 A-coded cable connector is supplied jointly with the BELS (it can also be ordered under the reference 202 237).
The cable must be shielded (refer to the data sheet of the assembly ANT80LP+ERL120).

Connection to the master unit: through a D-Sub 9-pin male socket; the shield is an option.



Annexe 5 : **Spécifications avec UML de classes du SEE**

+ = publique
- = privé

Localisateur

```
- parcoursCourant : table du référentiel
- struct  SPositionAbsolue
  {       avantDerniereEtiquette : char [4]
          dernieretiquette       : char [4]
          PositionRelative       : char [8]
  }
- trameLocalisation : struct  STrameLocalisation
   {      deb              : char                    //  début de trame STX
          etatTramway      : char [20]
          sensParcours     : char                    //  0=AR, 1=AV, 3=Inconnu
          nomArret         : char [20] ;             // RRR si en ligne ou hors ligne
          PositionAbsolue          : struct SPositionAbsolue
          vitesseLineaire  : char[7]
          tempsRoulage     : char[6]
          tempsImmobilisation : char[6]
          fin              : char                    //  fin de trame ETX
   }
// etatTramway : en terminus départ, en entrée de station, en ligne, en sortie de station, en terminus
// arrivée, hors ligne, en panne
- odom :                  Odometre *
- lecteurEtiq :           LecteurEtiquettes *
- esDeport :              ESDeportees *
```

```
+ ControlerLocalisation( ) // gère l'ordonnancement des différents traitements de localisation en
fonction des événements et des états d'un tramway
+ LocaliserTramway ( )                          // renseigne trameLocalisation
+ EnvoyerLocalisationTramway (struct  STrameLocalisation *)    // envoie trameLocalisation
+ FournirEtatTramway( ) : EtatsTramway
+ FournirNomArret() : char [20]
+ FournirVitesse () : reel                      // vitesse linéaire en m/s
+ FournirPositionAbsolue(SPositionAbsolue*)
+ FournirPositionRelative() : reel              // en m
+ PositionnerArret()                            // marque un arret dans parcoursCourant
+ FournirTempsRoulage( ) : entier               // en s
+ FournirTempsImmobilisation( ) : entier        // en s
```

Odometre

```
- serieBoitierOdo : CPortSerie *
- trameBrute :  char[29]            // pour stocker la trame reçue du module odomètre
- donneesOdometre : struct  SDonneesOdometre
{
        positionOdometrique     : float   // en m
        vitesseLineaire         : float   // en m/s
        tempsRoulage            : int     // en s
        tempsImmobilisation     : int     // en s
}
- diametreRoue          : short   // en mm
- nbCreneaux            : short
```

```
 - extraireDonneesOdometre()     : void  // renseigne donneesOdometre

+ RAZ()          : void
+ Demarrer()     : void
+ Arreter()      : void
+ Etalonner ()   : void
+ Recaler ()     : void
+ DelivrerDonneesOdo  (SdonneesOdometre  *) : void
```

LecteurEtiquettes

```
- etatDerniereLecture : enum etatLecture_t
      {LECT_OK, ERR_TRAME, ERR_BALOGH, ERR_NOTAG, ERR_MSG, ERR_INIT}
- numeroEtiquette : unsigned short        // unsigned short pour mot de 16 bits non signé
- dateLecture : time_t
- liaisonBalogh : fstream                 // liaison série avec lecteur Balogh
- pThreadRecevoirBalogh : Thread*
```

```
- Char2NumeroEtiquette (cfort : char, cfaible : char) : unsigned short // poids fort, poids faible
- RecevoirBalogh() : void          // fournit les valeurs de etatDerniereLecture et numeroEtiquette
- RecevoirTrame(tampon : char *) : bool

+ FournirNumeroEtiquette( pNumEtiquette : unsigned short *, pDateLecture : time_t * ) :
etatLecture_t
```

Annexe 6 : **Noyau temps réel (NTR++)**

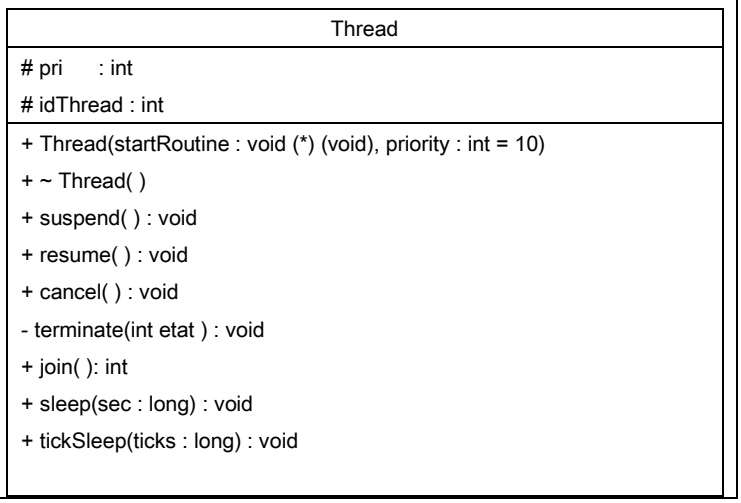
L'UCE est équipé d'un noyau temps réel objet multi-threads romable NTR++. Les classes Thread, Sémaphore et Event sont utilisables. On trouvera ci-dessous le fichier d'entête correspondant ainsi que la description UML.

Les Threads

```
/************************************************** Module Name:      Thread.h      ************************************************/

#ifndef  _THREAD_H
#define  _THREAD_H

/* Class Thread is the basic class for NTR++ thread control.
Usage example:
#include <Thread.h>
void threadEntryPoint( ) ;
void main( )
{ ….
Thread first(threadEntryPoint);
Thread second(threadEntryPoint, 50);
…
 first.cancel( ) ;
second.cancel( ) ;
}
--*/
```



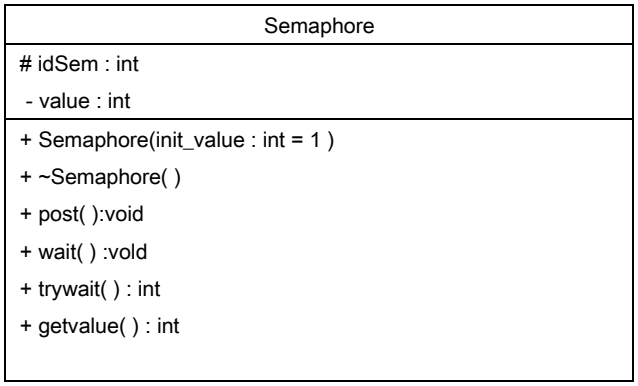

```
class Thread  {        // The scheduler is based on the highest-priority ready task determined  by two things :
                       // 1) the priority level you assign to the task when it is created
                       // 2) the order tasks are made ready among equal-priority tasks
 public:               // create and start a thread with  l(low) =<priority =<255(high)  (default=10)
      Thread (void (*startRoutine) ( ), int priority = 10);
       ~Thread ( );                  // delete the thread
      void   suspend ( );            // suspend thread
      void   resume ( );             // resume thread
      void   cancel ( );             // cancel thread
      static void    terminate ( );              // terminate current thread
      static void    sleep (long sec);           // delay current thread
      static void    tickSleep (long ticks);     // sleep in timer tick units (1/100e s)
protected:
      int    pri;                   //thread priority
      int    idThread;              //thread id
} ;
#endif   /* _THREAD_H */
```

Les Sémaphores

```
/****************************************** Module Name:        Semaphore.h    *********************************************/

#ifndef  _SEMAPHORE_H
#define  _SEMAPHORE_H

/* Class Semaphore is interface to the NTR++ semaphore object.
Usage example:
#include <Semaphore.h>
Semaphore s;              // binary semaphore
s.wait( );
 // critical section code here
s.post( );
-----------------------------------------------------------------
Semaphore sem(2); //general sémaphore,
   // can synchronize access to two resources units
sem.wait( );   //get one of the two resources units
//use the resource
sem.post( );
```




```
--*/

class Semaphore {                                :

public:
     Semaphore (int init_value = 1);
     ~Semaphore ();
     void post( );            //increment semaphore V(s)
     void wait ( );            //decrement semaphore (with blocking if sem<0) P(s);
     int trywait ( );          // ifsem >0 decrement semaphore P(s) and return 0, if sem =0 return ER_NMP without blocking
     int getvalue ( );         // return semaphore value
protected:
     int idSem;              //semaphore id
private:
     int value;              //semaphore value
} ;
#endif     /* _SEMAPHORE_H */
```

Les Evénements

```
/************************************************** Module Name:  Event.h  ****************************/

/* SANS OBJET POUR LE SUJET */
```

Annexe 7 : **Afficheur SX502**

1. Caractéristiques techniques :

La version de l'appareil est codée comme suit dans la désignation du type :

SX502 – [ ][ ]0 / 0[ ] / [ ][ ] – [ ][ ][ ] / [ ][ ] – M0

2 lignes	2									
4 lignes	4									
6 lignes	6									
8 lignes	8									
20 caract./ligne*		2								
40 caract./ligne*		4								
Hauteur de caractères 33/66/75 mm			3							
Hauteur des caract. 50/100/120 mm			5							
LED standard				0						
LED pour applications extérieures				2						
Couleur des LED rouge					R					
Couleur des LED commutable rouge/vert/orange					M					
Affichage monoface						1				
Affichage bi-faces						2				
Boîtier tôle d'acier, laqué							0			
Boîtier tôle d'acier, laque double couche							1			
Boîtier acier inoxydable V2A, laqué							2			
Boîtier acier inoxydable V2A, brossé							3			
Boîtier acier inoxydable V4A, brossé							5			
Indice de protection IP54								0		
Indice de protection IP65								1		
Indice de protection IP54 avec compensation climatique								2		
Indice de protection IP54 avec compensation climatique et chauffage								4		
Montage mural, entrée de câble en bas									0	
Montage mural, entrée de câble en haut									1	
Montage suspendu, entrée de câble en bas									2	
Montage suspendu, entrée de câble en haut									3	
Montage mural et suspendu, entrée de câble en bas									4	
Montage mural et suspendu, entrée de câble en haut									5	
Tension d'alimentation 230 V AC ±15 %, 50 Hz										A
Tension d'alimentation 24 V DC ±15 %										B
Tension d'alimentation 115 V AC ±15 %, 60 Hz										C

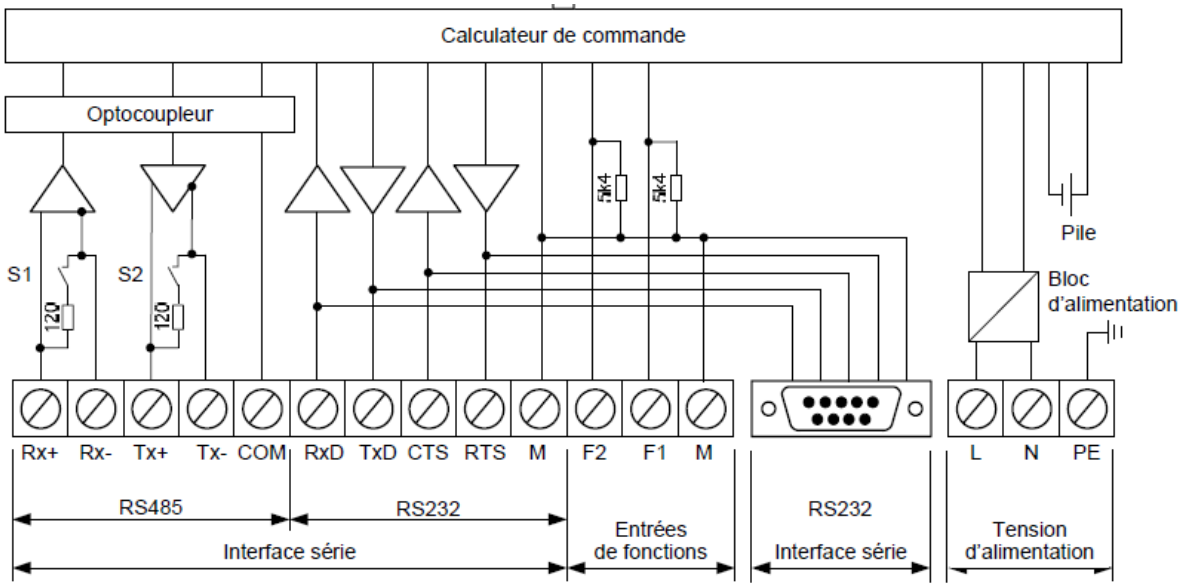
* concerne la fonte Acala 7

2. Affichage des caractères

La largeur des caractères est de 5 pixels et l'espacement entre les caractères est de 1 pixel.
La hauteur des caractères est de 7 pixels.
L'interligne est de 2 Pixels.

Fonte de caractères	SX502-xxx/03/xx-xxx/xx-xx	SX502-xxx/05/xx-xxx/xx-xx
Acala 7	env. 33 mm	env. 50 mm
Acala 7 extended	env. 33 mm	env. 50 mm
Acala 14 condensed	env. 66 mm	env. 100 mm
Acala 14	env. 66 mm	env. 100 mm
Acala 14 extended	env. 66 mm	env. 100 mm
Acala 16 condensed	env. 75 mm	env. 120 mm
Acala 16	env. 75 mm	env. 120 mm
Acala 16 extended	env. 75 mm	env. 120 mm

3. Schéma de principe

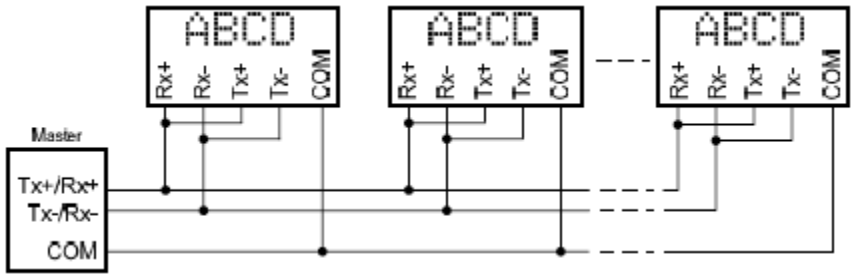


4. Interface

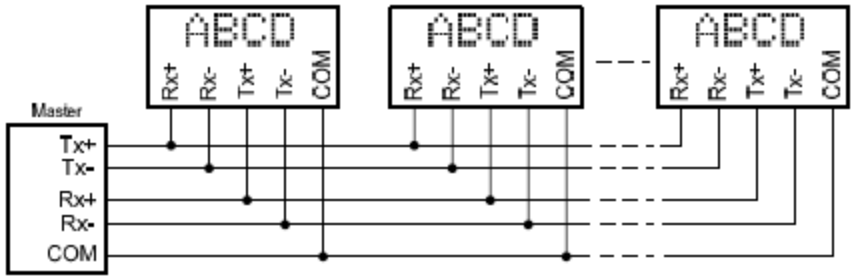
Pour la commande Modbus, utiliser l'interface RS485.
L'interface RS232 n'est pas recommandée pour la commande Modbus. Elle est prévue pour la programmation de l'appareil avec un PC, par exemple pour charger des textes statiques dans la mémoire de textes et pour installer des fontes de caractères à l'aide de l'outil de PC 'DisplayManager', livré sur supports de données.

5. Câblage et raccordement

La documentation *Modbus over serial line specification and implementation guide* recommande le câblage de raccordement d'un bus RS485 2 fils (Two-Wire Modbus Definition) comme représenté sur le schéma page suivante.



Il est également possible de réaliser le câblage de raccordement d'un bus RS485 4 fils (Optional Four-Wire Modbus Definition) comme représenté sur le schéma ci-dessous.



Les lignes de données de l'interface RS485 doivent être équipées aux deux extrémités de résistances de terminaison pour obtenir un fonctionnement sans perturbations. Les résistances nécessaires se trouvent sur le calculateur de commande et peuvent être activées avec les commutateurs S1 pour Rx et S2 pour Tx.

6. Commande de l'afficheur

6.1. Modbus

Les appareils sont commandés en esclave Modbus RTU (Remote Terminal Unit).
Ils utilisent le code fonction 16 (0x10) : Ecriture de registres multiples (Write Multiple Registers) conformément au protocole Modbus.

6.2. Mode d'opération 'textes dynamiques et statiques'

Adresse de départ
Pour l'adresse de départ, utiliser l'adresse de registre 0000h.

Interprétation des données
Les données sont interprétées selon le tableau des commandes suivant. Dans la description des commandes, les chiffres indiqués entre [ ] se rapportent aux lignes correspondantes du tableau des commandes.

Tableau des commandes :

Commandes de manipulation des textes

Afficher texte dynamique	`cc...↵`	Envoyer des caractères quelconques	[1]
Afficher texte statique	`$Tn↵`	Appeler texte statique (n = numéro de texte, 1 à 4 digits)	[2]
Afficher texte dynamique	`$Lxxcc...↵`	Envoyer des caractères quelconques à la ligne xx	[43]

Dans les lignes [1] et [43], **cc...** représente une chaîne de caractères avec un contenu quelconque.

Ligne [43] : Les lignes sur l'afficheur sont numérotées à partir de 1.

Commandes de formatage de textes

Retour à la ligne	`$C`	Retour à la ligne forcé	[6]
Clignotement	`$F1`	Clignotement des caractères suivants marche	[7]
	`$F0`	Clignotement des caractères suivants arrêt	[8]
Texte déroulant	`$Y`	Texte déroulant à partir de la position actuelle jusqu'à la fin du texte ou $C	[9]
Fonte de caractères	`$M1`	Acala 7	[10]
	`$M2`	Acala 7 extended	[11]
	`$M3`	Acala 14 condensed	[12]
	`$M4`	Acala 14 extended	[13]
	`$M5`	Acala 7 P / Fonte de caractères personnalisée 7 pixels	[14]
	`$M6`	Acala 14 / Fonte de caractères personnalisée 14/16 pixels	[15]

Annexe 8 : **Modbus**

1. Modbus

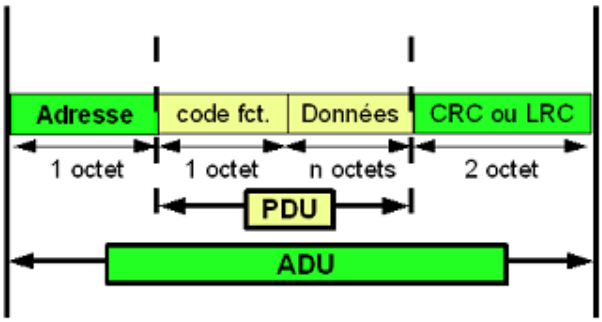
1.1. Présentation

MODBUS est un protocole de communication couramment utilisé dans l'industrie pour faire communiquer des automates programmables.
Le protocole MODBUS est un protocole de dialogue basé sur une structure hiérarchisée entre un maître et plusieurs esclaves. Les esclaves possèdent une adresse comprise entre 1 et 64.

1.2. Échange Maître/ Esclave sur Modbus

Le maître interroge un esclave de numéro unique sur le réseau et attend de la part de cet esclave une réponse.
Lorsque l'esclave envoie sa réponse, il place sa propre adresse dans le champ adresse afin que le maître puisse l'identifier.

1.3. Format d'une trame Modbus



Une trame Modbus est composée de deux parties :

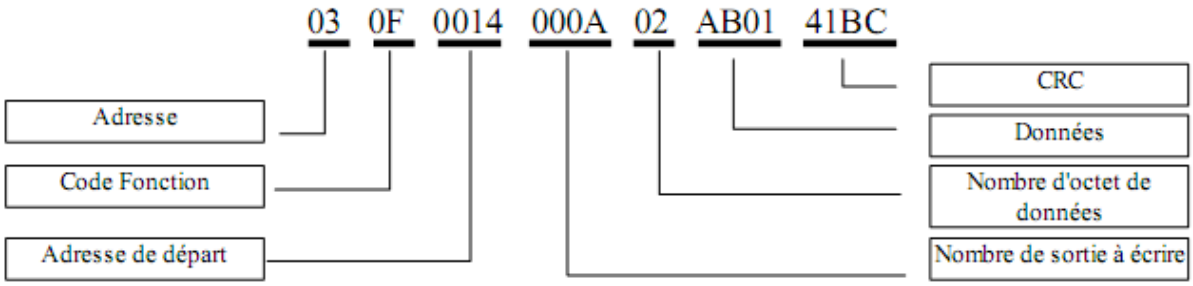
- **"ADU" Application Data Unit** : Cette partie est fonction de la couche de communication sur laquelle s'appuie Modbus.
  L'exemple ci-dessus correspond à un ***ADU Modbus sur liaison série***.
  Pour une trame ***ModbusTCP l'ADU*** est différent (voir le paragraphe 3 de cette annexe).
- **"PDU" Protocol Data Unit** : Il est composé des champs :
  - ***"Code fonction"*** : Ce champ est prédéfini par le protocole Modbus (voir paragraphe 2 de cette annexe). Ce code fonction définie l'action à exécuter sur l'esclave.
    Par exemple le code fonction 0x0F (write Multiple Coils) permet de modifier les sorties numériques sur un esclave.
  - ***"Données"*** : Dans le champ "Données" sont présentes des informations relatives au code fonction (exemple : adresse de registre pour l'écriture sur les sorties) et des données à échanger entre le maître et l'esclave (exemple : les informations relatives aux sorties à mettre à l'état haut).

Codage des trames Modbus sur liaison série :

Deux types de codage peuvent être utilisés pour communiquer sur un réseau Modbus sur liaison série : ASCII ou RTU

- Mode ***ASCII*** : Chaque octet composant une trame est codé avec 2 caractères ASCII (2 fois 8 bits).
  Par exemple, la valeur 01 est codée avec les codes ASCII des symboles '0' et '1' (soit 0x30 et 0x31).
- Mode ***RTU*** : Les octets composants la trame ne sont pas codés.
  Par exemple, la valeur $(15)_{10}$ aura pour octet 0x0F ou $(00001111)_2$.

Exemple d'une requête modbus RTU pour écrire une série de 10 sorties commençant à l'adresse 20 sur l'esclave à l'adresse 3:



2. Fonctions Modbus

15 (0x0F) Write Multiple Coils

This function code is used to force each coil in a sequence of coils to either ON or OFF in a remote device. The Request PDU specifies the coil references to be forced. Coils are addressed starting at zero. Therefore coil numbered 1 is addressed as 0.
The requested ON/OFF states are specified by contents of the request data field. A logical '1' in a bit position of the field requests the corresponding output to be ON. A logical '0' requests it to be OFF.
The normal response returns the function code, starting address, and quantity of coils forced.

Request PDU

Function code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0001 to 0x07B0
Byte Count	1 Byte	**N***
Outputs Value	**N*** x 1 Byte	

***N** = Quantity of Outputs / 8, if the remainder is different of 0 ⇒ N = N+1

Response PDU

Function code	1 Byte	**0x0F**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Outputs	2 Bytes	0x0001 to 0x07B0

Error

Error code	1 Byte	**0x8F**
Exception code	1 Byte	01 or 02 or 03 or 04

16 (0x10) Write Multiple registers

This function code is used to write a block of contiguous registers (1 to 123 registers) in a remote device.
The requested written values are specified in the request data field. Data is packed as two bytes per register.
The normal response returns the function code, starting address, and quantity of registers written.

Request

Function code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	0x0001 to 0x007B
Byte Count	1 Byte	2 x **N***
Registers Value	**N*** x 2 Bytes	value

***N** = Quantity of Registers

Response

Function code	1 Byte	**0x10**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 123 (0x7B)

Error

Error code	1 Byte	**0x90**
Exception code	1 Byte	01 or 02 or 03 or 04

3. MobusTCP

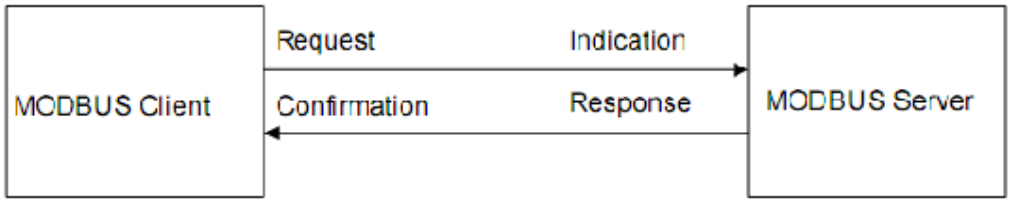
3.1. Présentation

ModbusTCP est la variante "encapsulée" dans TCP/IP du protocole Modbus.
Le protocole ModbusTCP est basé sur un mode de communication client / serveur, celui-ci permet d'échanger des informations entre différents produits d'automatismes sur Ethernet TCP/IP.

Il existe plusieurs familles d'équipement ModbusTCP :

- Des clients et des serveurs Modbus TCP.
- Des matériels d'interconnexion (ponts, routeurs ou passerelles) assurant le lien entre le réseau TCP/IP et une ou plusieurs liaisons Modbus Série, sur lesquelles sont connectés différents terminaux Modbus.

3.2. Le modèle de ModbusTCP



Le modèle Modbus TCP est basé sur 4 types de message :

- Modbus request : Message envoyé par un client sur le réseau pour initier une transaction
- Modbus Indication : c'est le message "Modbus request" reçu par le serveur modbusTCP
- Modbus Response : Message de réponse envoyé par le serveur
- Modbus Confirmation : C'est le message de réponse reçu par le client.

3.3. Encapsulation d'une trame Modbus dans ModbusTCP

Le mapping du protocole Modbus sur des bus spécifiques ou des réseaux peut introduire des champs supplémentaires au niveau de l'unité de donnée d'application, ou ADU pour Application Data Unit. Pour ModbusTCP, le champ supplémentaire ajouté dans l'ADU est appelé : MBAP (Modbus Application Protocol Header)
Description du champ MBAP :

- **Transaction Identifier** (2 octets): il s'agit en fait du numéro d'un message MODBUS. Chaque message (requête+réponse) est identifié par un numéro différent. Ce numéro est initialisé par le client et le serveur le recopie dans la réponse.
- **Protocol Identifier** (2 octets): Numéro du protocole pour les systèmes multiplexés qui en utilisent plusieurs (MODBUS= 0).
- **Length** (2 octet) : Longueur du message à suivre (Unit Identifier + Data)
- **Unit Identifier** (1 octet) : Adresse de l'esclave connecté en liaison série derrière une passerelle ModbusTCP / Modbus

Annexe 9 :**Réseau Multi Service (RMS)**

CODEC	Bande passante données audio	Période échantillonnage	Taille échantillon (octets)	Nb d'échantillons par paquet
G.711 (PCM)	64 kbps	30 ms	240	1
G.726 (ADPCM)	32 kbps	20 ms	80	1
G.728 (LD-CELP)	16 kbps	2.5 ms	5	4(*)
G.729a (CS-CELP)	8 kbps	10 ms	10	2(*)

(*) : Lorsque les paquets regroupent chacun **n** échantillons, la période d'émission des paquets est de **n** fois la période d'échantillonnage. Ex : en G.728, la période d'émission des paquets est de 4x2,5ms = 10ms.

Tableau An.10.1 : codecs et constitution des paquets VoIP

Débit	Norme	Câble	Longueur maxi	Coût relatif
Ethernet	10BASE5	Thick Ethernet (coax)	500m	obsolète
	10BASE-T	Twisted Pair (cat 3)	100m	Très faible
Fast Ethernet	100BASE-TX	Twisted Pair (cat 5)	100m	Très faible
	100BASE-FX	Multimode Optical Fiber	412m (half duplex)	moyen
		Multimode Optical Fiber	2000m (full duplex)	moyen
		Singlemode Optical Fiber	15km (full duplex)	élevé
Gigabit Ethernet	1000BASE-T	Twisted Pair (cat 5e ou 6)	100m	faible
	1000BASE-SX	Multimode Optical Fiber	550m	moyen
	1000BASE-LX	Singlemode Optical Fiber	5km	élevé
	1000BASE-LH	Singlemode Optical Fiber	Jusqu'à 70km	Très élevé
10 Gigabit Ethernet	10GBASE-LX4	Multimode Optical Fiber	300m	Très élevé
	10GBASE-ER/EW	Singlemode Optical Fiber	jusqu'à 40km	extrêmement élevé

Tableau An.10.2 : Caractéristiques des supports Ethernet

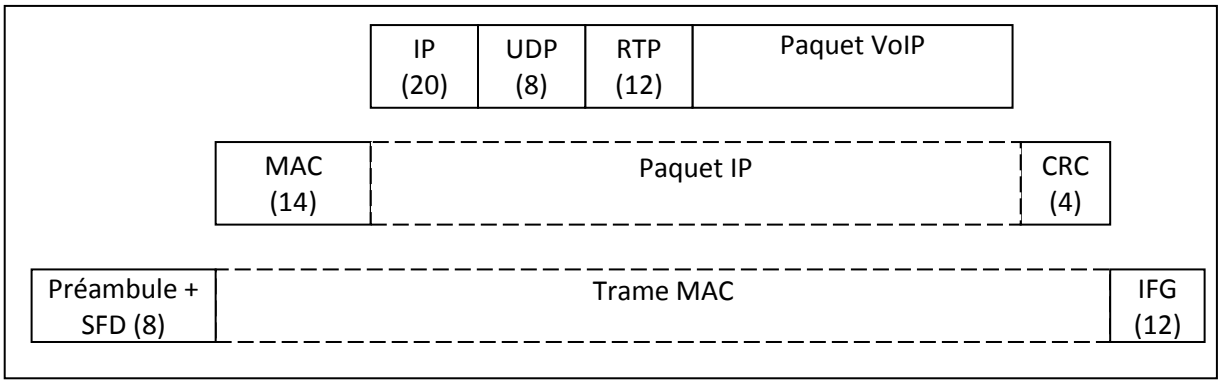


Figure An.10.3 : encapsulation VoIP sur Ethernet (Nb octets entre parenthèses)