Chapitre **Page**

Ce que vous trouverez dans ce manuel

Chapitre 1 : **Présentation**

à quoi sert le produit, de quoi il est composé, la liste des fonctions disponibles, le principe d'utilisation d'une fonction, la description des principales fonctions, les caractéristiques et performances.

Chapitre 2 : **Installation**

environnement requis, procédure d'installation, organisation des fichiers résultants, modification des paramètres de configuration.

Chapitre 3 : **Utilisation**

les différentes étapes pour développer une application utilisant les services de la librairie TPUNI-TE, un exemple d'application qui peut servir à tester la liaison avec un équipement distant.

Chapitre 4 : **Syntaxe des fonctions**

syntaxes à respecter pour appeler une fonction.

Chapitre 5 : **Annexes**

Lexique des termes utilisés, liste des codes d'erreurs.

A qui s'adresse ce manuel

Ce document s'adresse à des développeurs d'application en langage C++.

Les prérequis

Nous supposons que l'utilisateur connaît déjà :
- le langage "C"
- le système d'exploitation DOS ou OS/2
- les objets UNI-TE des équipements

Documentation Telemecanique associée

TSX DM FPP F	: Mise en œuvre liaison bus FIPIO sur processeur TSX/ PMX modèle 40 version V6.
TSX DM PCX V52F	: Manuel de mise en œuvre du coprocesseur PL7-3.

Sommaire

1.1 But du logiciel TP UNI-TE

Le logiciel TP UNI-TE permet la connexion de micro-ordinateurs compatibles PC sous DOS, OS/2 et WINDOWS avec un coprocesseur PL7-3.

Elle permet une mise en œuvre simplifiée et fiable de la communication entre un PC et des automatismes Telemecanique, sans nécessiter une connaissance spécifique de ceux-ci.

Intégration d'applications dans l'architecture de communication X-WAY

L'objectif des réseaux de communication normalisés ISO/OSI est de permettre l'élaboration d'architectures d'équipements de provenance et de nature très diverses, qu'il s'agisse de l'informatique de planification et de gestion de production ou de contrôleurs d'automatismes tels qu'automates programmables, commandes numériques de machines outils ou de robots, système de vision etc...

L'architecture de communication X-WAY, conforme au modèle OSI, comprend un ensemble de réseaux locaux industriels permettant la construction de structures d'automatismes répartis, des plus simples aux plus complexes.

L'offre logicielle TP UNI-TE est mise en œuvre sur des micro-ordinateurs exécutant par exemple :

- des applications autonomes de production ou de traitements de données
- des application passerelles entre le monde des automatismes et celui de la gestion, planification.
- .....

Rappel sur UNI-TE :

Le système de messagerie industriel Telemecanique supporté par l'architecture de communication X-WAY s'appelle UNI-TE.
Le protocole UNI-TE fonctionne suivant un mécanisme Client/Serveur de Requêtes/ Réponses (services confirmés).
Un équipement supportant le protocole UNI-TE peut être :

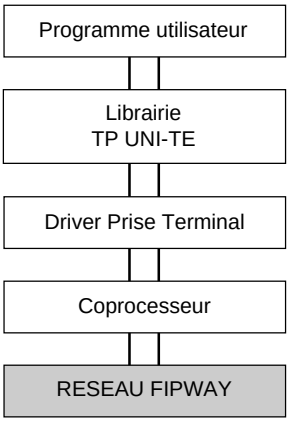
- CLIENT : c'est l'équipement qui prend l'initiative de la communication avec un autre équipement SERVEUR, il récupère (lecture) ou transmet (écriture) des informations ou envoie un ordre (Run, Stop)
- SERVEUR : c'est l'équipement qui rend le service demandé par le CLIENT et lui envoie une réponse après exécution.

Un équipement peut émettre un message de sa propre initiative vers un autre équipement sans que le message ne soit confirmé. Les données émises s'intitulent alors données non sollicitées (DNS).

Le micro-ordinateur équipé du logiciel TP UNI-TE est un utilisateur UNI-TE de type CLIENT.

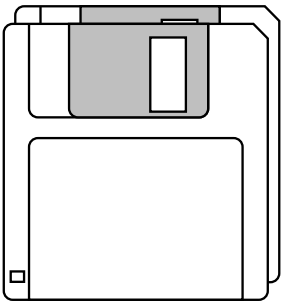
Le logiciel TP UNI-TE est une "librairie" de fonctions écrites en langage C.
Elle permet à des programmes utilisateurs d'exploiter de façon simplifiée la communication en s'affranchissant des connaissances nécessaires à la mise en œuvre des différentes couches logicielles qui la composent.

Cette librairie vient s'intercaler entre le programme utilisateur et le driver TSXPC dont elle exploite les possibilités de communication.

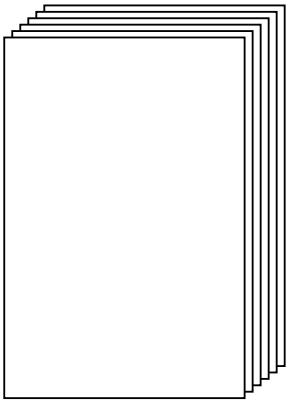


Les principaux services offerts par TP UNI-TE sont les suivants :
- Gestion du contexte de communication.
- Gestion des équipements
- Accès aux variables
- Réception de données non sollicitées
- Gestion de domaine
- Fonction générique
- Récupération des réponses

1.2 Composition du produit



Deux disquettes 3"1/2



La présente documentation avec licence

Le logiciel TP UNI-TE livré, a été développé en langage Microsoft Visual C++.

1.3 Principe d'utilisation

1.3-1 Définition de termes et de fonctionnalités

Pour comprendre le principe d'utilisation de la librairie, il est nécessaire de définir quelques termes.

Client UNI-TE : équipement capable d'émettre des requêtes UNI-TE.

Serveur UNI-TE : équipement répondant aux demandes de services (requêtes) émises par un équipement client.

Le programme utilisateur est client UNI-TE c'est-à-dire qu'à son initiative, il accède aux équipements serveurs UNI-TE du réseau par exploitation des services de la librairie listés dans les tableaux du chapitre 1.5.

Du fait de cette organisation CLIENT/SERVEUR, l'équipement de 2 PC avec 2 logiciels TP UNI-TE ne permet pas de les faire communiquer entre eux.

ADRESSAGE 5 NIVEAUX XWAY : dans le système d'adressage TELEMECANIQUE, les adresses des émetteurs et destinataires sont codées sur 5 octets :
- **réseau :** numéro de réseau.
- **station :** numéro de l'équipement sur le réseau.
- **porte :** identification des entités logiques de la station.

Si l'équipement est un automate programmable et que le numéro de porte est égal à 5:
- **module :** emplacement géographique d'un coupleur dans l'automate.
- **voie :** numéro d'une entité logique du coupleur ou adresse d'un esclave UNI-TELWAY + 100.

Pour l'accès au serveur UNI-TE du coprocesseur PL7-3 :

(réseau)	=	0
(station)	=	254
(porte)	=	0
(module)	=	0
(voie)	=	0

Pour l'accès à un bloc TxT n de l'application PL7-3 du coprocesseur :

(réseau)	=	0
(station)	=	254
(porte)	=	16 + 11
(module)	=	0
(voie)	=	0

Pour accéder à un ou à des équipements distants, il est nécessaire d'ouvrir un ou plusieurs canaux de communication :

Canal de communication : ressource permettant d'émettre les requêtes (liées à des fonctions) à destination d'un équipement distant.

Le logiciel TP UNI-TE peut gérer jusqu'à 3 canaux de communication.

Un canal de communication ne peut émettre des requêtes qu'à destination d'un seul

équipement serveur.

Il est possible d'ouvrir plusieurs canaux de communication à destination du même équipement.

Sur un canal de communication une requête n'est émise que si la réponse à la précédente est parvenue.

Plusieurs requêtes peuvent être émises simultanément en utilisant des canaux différents.

1.3-2 Utilisation

Pour communiquer avec un équipement à partir d'une application PC, le programme utilisateur doit suivre les étapes suivantes :

1 . initialiser le contexte de communication :
utilisation de la fonction *UNITEInitdriver*

2 . ouvrir un ou plusieurs canaux de communication à destination de un ou plusieurs équipements.
utilisation de la fonction *UNITEOpenConnection* qui retourne un numéro de canal *hEquip.*

3 . gérer la séquence suivante à chaque accès :
- utiliser une fonction *UNITE_xx_xx* en indiquant le canal dédié (hEquip), l'endroit où on désire stocker la réponse, et les données utiles correspondant à la fonction.
- la fonction retourne un Identificateur de requête *hReq*
- poursuivre son traitement
- émettre la demande de réponse à la fonction précédemment utilisée en utilisant la fonction UNITEResponse avec l'identificateur *hReq*
- si les données sont prêtes, la fonction retourne la réponse OK indiquant que les données stockées à l'adresse prédéfinie sont valides.

lorsque les accès sont terminés :

4 . fermer le ou les canaux de communication.
utilisation de la fonction *UNITECloseConnection*

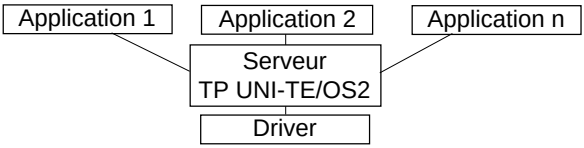
5 . libérer le contexte de communication :
utilisation de la fonction *UNITECloseDriver*

Le programme utilisateur peut recevoir des données non sollicitées. Un canal de communication prédéfini (porte 3) peut être ouvert en réception pour recueillir ce type de données.

L'application arrivera à envoyer des données non sollicitées par l'initialisation d'un bloc TXT de type TERminal dont le paramètre Txti,t sera positionné à 3 par le programme application.

1.3-3 Complément d'utilisation sous OS/2

Le système d'exploitation OS/2 est multitâche et permet à plusieurs programmes application de faire appel aux services de la librairie de façon asynchrone.



1.4 Description des principales fonctions

1.4-1 Gestion du contexte de communication

Les fonctions de gestion du contexte de communication permettent l'utilisation du logiciel TP UNI-TE à partir d'une ou de plusieurs applications.

UNITEInitDriver :	initialisation du contexte de communication
UNITECloseDriver :	libération du contexte de communication

Ces 2 fonctions doivent être activées au début (***UNITEInitDriver***) et à la fin (***UNITECloseDriver***) de chaque application de communication.

UNITEOpenConnection :	ouverture d'un canal de communication
UNITECloseConnection:	fermeture d'un canal de communication

Avant de pouvoir émettre des requêtes à destination d'un équipement distant, il est nécessaire d'ouvrir un ou plusieurs canaux de communication.
Plusieurs canaux peuvent être ouverts à destination d'un même équipement.
On ne peut activer qu'une seule fonction à la fois par canal.
Lorsqu'on a terminé d'utiliser un canal, on le referme.

UNITEOpenUnsollicitedData :	ouverture du canal de réception de données non sollicitées
UNITECloseUnsollicitedData :	fermeture du canal de réception de données non sollicitées

Avant de pouvoir recevoir des données non sollicitées d'un équipement distant, il est nécessaire d'ouvrir un canal en réception. Les émetteurs de DNS à destination du PC doivent utiliser l'adresse XWAY destinataire suivante : Réseau et station par défaut (0 : RES, 254 : station), Porte égale à 3, Module et Voie à 0 ou par bloc TXT de type TER LOCAL avec Txti,t = 3 dans le programme, Module et Voie à 0.

UNITETimeOut :	réglage du temps d'attente maximum

Lorsque le logiciel TP UNI-TE émet une requête à destination d'un équipement distant, il lance une temporisation égale à ce temps.
Si dans le programme application, on active la fonction ***UNITEResponse*** et que cette temporisation est arrivée à terme sans qu'on ait obtenu de réponse, UNITEResponse renverra un code d'erreur TIME_OUT.

1.4-2 Gestion des équipements

Ces différentes fonctions permettent de tester le chemin de communication avec un équipement distant, d'obtenir l'identification d'un équipement distant, de le mettre en marche ou à l'arrêt, de gérer les mécanismes de réservation-déréservation, etc.

1.4-3 Accès aux données

Ces fonctions permettent d'accéder en lecture ou en écriture aux objets UNI-TE situés dans le coprocesseur ou un équipement distant.

1.4-4 Données non sollicitées

Cette fonction permet de récupérer des données non sollicitées émises par un équipement distant ou le coprocesseur.

1.4-5 Fonction générique

Cette fonction permet de générer n'importe quelle requête UNI-TE.

1.4-6 Récupération des réponses

La fonction ***UNITEResponse*** permet de récupérer la réponse à une requête précédemment envoyée.
La réponse peut être pollée ou attendue avec un temps d'attente infini.

1.5 Tableaux récapitulatifs des fonctions

Les tableaux suivants récapitulent l'ensemble des fonctions disponibles classées par type de service.

NOM DE LA FONCTION	GESTION DU CONTEXTE DE COMMUNICATION
UNITEInitDriver	initialisation du contexte de communication
UNITECloseDriver	libération du contexte de communication
UNITEOpenConnection	ouverture d'un canal de communication
UNITECloseConnection	fermeture d'un canal de communication
UNITEOpenUnsollicitedData	ouverture du canal de communication en réception de données non sollicitées
UNITECloseUnsollicitedData	fermeture du canal de communication en réception de données non sollicitées
UNITETimeOut	réglage du temps d'attente maximum
UNITEGetSystemError (seulement OS/2)	récupération d'une erreur système OS/2

NOM DE LA FONCTION	GESTION DES EQUIPEMENTS
UNITEMirror	test de la connexion d'un équipement
UNITEIdentification	identification d'un équipement
UNITEReserve	réservation d'un équipement
UNITERelease	déréservation d'un équipement
UNITEIAmAlive	entretien de réservation
UNITERun	mise en marche d'un équipement
UNITEStop	arrêt d'un équipement

NOM DE LA FONCTION	ACCES AUX VARIABLES
UNITEReadInternalBit	lecture variable de type bit interne
UNITEReadSystemBit	lecture variable de type bit système
UNITEWriteInternalBit	écriture variable de type bit interne
UNITEWriteSystemBit	écriture variable de type bit système
UNITEReadInternalWord	lecture variable de type mot interne
UNITEReadConstantWord	lecture variable de type mot constant
UNITEReadSystemWord	lecture variable de type mot système
UNITEWriteInternalWord	écriture variable de type mot interne
UNITEWriteSystemWord	écriture variable de type mot système
UNITEReadCommonWord	lecture variable de type mot commun
UNITEWriteCommonWord	écriture variable de type mot commun
UNITEReadInternalDWord	lecture variable de type double mot interne
UNITEReadConstantDWord	lecture variable de type double mot constant
UNITEWriteInternalDWord	écriture variable de type double mot interne
UNITEReadObject	lecture variable de type objet
UNITEWriteObject	écriture variable de type objet
UNITEReadWordArray	lecture variable de type table de mots internes
UNITEWriteWordArray	écriture variable de type table de mots internes

NOM DE LA FONCTION	GESTION DE DOMAINE
UNITETransferTsxPC	télédéchargement du programme automate
UNITETransferPCTsx	téléchargement du programme automate

NOM DE LA FONCTION	DONNES NON SOLLICITEES
UNITEUnsollicitedData	récupération de données non sollicitées

NOM DE LA FONCTION	FONCTION GENERIQUE
UNITERequest	fonction permettant l'émission de tout type de requête UNI-TE

NOM DE LA FONCTION	RECUPERATION DES REPONSES
UNITEResponse	récupération de la réponse à une requête
UNITEResponseMult (seulement OS/2)	récupération de la première réponse parmi une liste de requêtes émises

Fonctions	**Valeurs**
Nombre de canaux de communication	3
Nombre de requêtes simultanées	3
Nombre de cartes PCX dans le PC	1
Nombre de données non sollicitées	1

2.1 Installation sous DOS

Procédure

Avant d'installer le logiciel TP UNI-TE sur le disque dur, il est conseillé de :

- lire le certificat de licence et de garantie concernant les restrictions de copie et d'installation de logiciel.
- dupliquer la disquette nécessaire à l'installation afin de la préserver contre toute détérioration accidentelle .

L'installation s'effectue en suivant les étapes suivantes (les commandes tapées au clavier sont en caractères gras et en italique) :

1. Insérer la disquette contenant la librairie TP UNI-TE dans le drive A: sous DOS

2. Taper la commande : ***a:*** puis appuyer sur la touche ***RETURN***
[a:\] apparaît à l'écran.

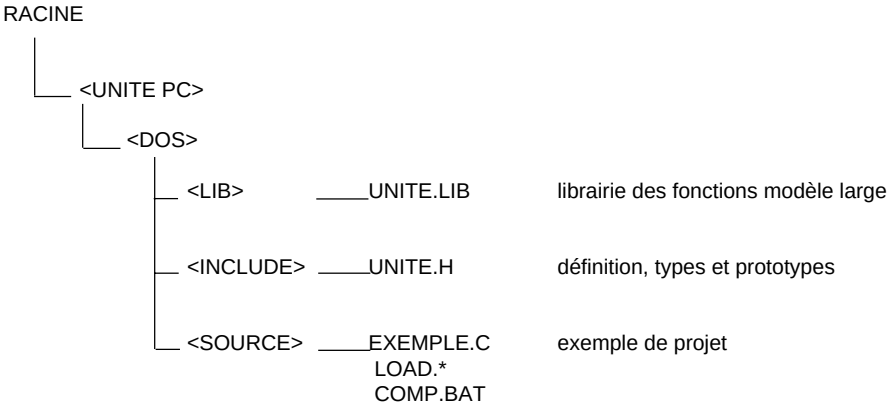
3. La commande "install destdrive" permet de ranger les fichiers de la librairie sur le disque souhaité.
Par exemple, pour les installer sur le disque C, on tape la commande :
install c, puis on appuie sur la touche ***RETURN***

Le fichier C:\AUTOEXEC.BAT est à modifier de la façon suivante :

SET INCLUDE = (...config. déjà existante ...); Disque:\UNITEPC\DOS\INCLUDE
SET LIB = (..configuration déjà existant); Disque:\UNITEPC\DOS\LIB

4. Rebooter votre système.

Organisation des fichiers résultants



2.2 Installation sous OS/2

Environnement requis
Micro-ordinateur compatible PC bus ISA (ou EISA) microprocesseur type 386 au minimum, sous système d'exploitation OS/2 version 2.1.

Procédure
Avant d'installer le logiciel TP UNI-TE sur le disque dur, il est conseillé de :
- lire le certificat de licence et de garantie concernant les restrictions de copie et d'installation de logiciel.
- dupliquer la disquette nécessaire à l'installation afin de la préserver contre toute détérioration accidentelle .

L'installation s'effectue selon les étapes suivantes (les commandes tapées au clavier sont en caractères gras et en italique) :

1. Insérer la disquette contenant la librairie TP UNI-TE/OS2 dans le drive A:

2. Sélectionner une fenêtre OS/2 plein écran

3. Taper la commande : ***a:*** puis appuyer sur la touche ***RETURN***
[a:\] apparaît à l'écran.

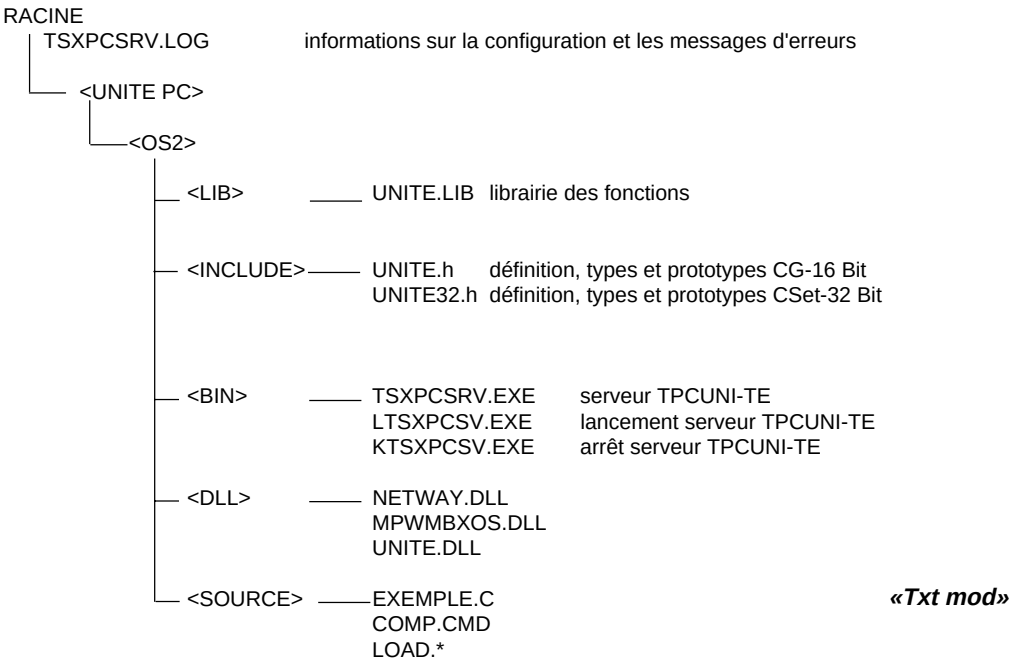
4. La commande "install destdrive" permet de ranger les fichiers de la librairie sur le disque souhaité, et génère un groupe de deux icônes qui permettent le lancement et l'arrêt du serveur.
Par exemple, pour les installer sur le disque C, on tape la commande :
install c, puis on appuie sur la touche ***RETURN***

Le fichier C:\CONFIG.SYS est à modifier de la façon suivante :
IOPL = yes
LIBPATH = (...configuration déjà existante) ; Disque:\UNITEPC\OS2\DLL
SET PATH = (...configuration déjà existante) ; Disque:\UNITEPC\OS2\BIN
SET INCLUDE = (...configuration déjà existante) ; Disque:\UNITEPC\OS2\INCLUDE
SET LIB = (...configuration déjà existante) ; Disque\UNITEPC\OS2\LIB

5. Rebooter votre système. Les paramètres de configuration du programme ont alors les valeurs par défaut suivantes :
NP : nombre d'application OS/2 gérables : 32
NU : nombre de données non sol. bufferisées : 10
Pour modifier ces paramètres, se reporter au paragraphe "Modifications" page 20.

6. Cliquer deux fois sur le groupe de programme : "Librairie UNI-TE PCX OS/2" puis cliquer deux fois sur l'icône "Lancement Serveur UNI-TE PCX OS/2".

Organisation des fichiers résultants



Modification des paramètres de configuration

Par exemple pour gérer 50 canaux de communication et 2 cartes coupleurs.

1. Cliquer deux fois sur l'icône "Arrêt Serveur UNITE PCX OS/2"
2. Cliquer avec le bouton droit de la souris sur l'icône "Lancement Serveur UNITE PCX OS/2"
3. Cliquer sur la flêche de l'item "Ouverture"
4. Cliquer la case en haut et à gauche de la fenêtre "Lancement Serveur UNITE PCX OS/2 - Paramètre"
5. Cliquer sur l'item "Arrêt/Fermeture"
6. Cliquer deux fois sur l'icône "Lancement Serveur UNITE PCX OS/2"

«Txt mod»

2.3 Installation sous WINDOWS

Environnement requis

Micro-ordinateur compatible PC bus ISA (ou EISA) microprocesseur type 386 au minimum, sous système d'exploitation WINDOWS 3.1x ou WINDOWS 95.

Procédure

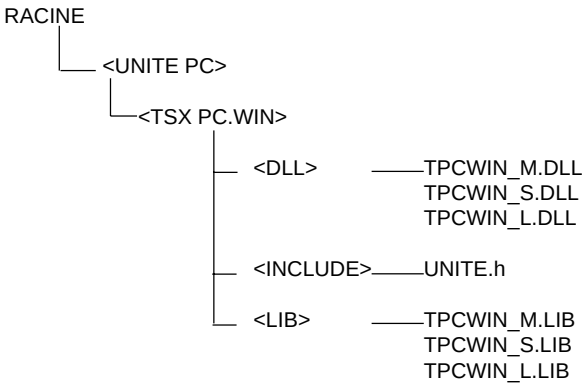
Avant d'installer le logiciel TP UNI-TE sur le disque dur, il est conseillé de :

- lire le certificat de licence et de garantie concernant les restrictions de copie et d'installation de logiciel.
- dupliquer la disquette nécessaire à l'installation afin de la préserver contre toute détérioration accidentelle .

L'installation s'effectue selon les étapes suivantes (les commandes tapées au clavier sont en caractères gras et en italique) :

1. Se placer sous l'environnement WINDOWS.
2. Insérer la disquette contenant la librairie TP UNI-TE/WINDOWS dans le drive A.
3. Puis sous WINDOWS, EXECUTER ***a: setup*** et suivre la procédure.
4. Si le driver PRISE TERMINALE n'est pas déjà installé, insérer la disquette TSX LF TP, exécuter ***a: setup,*** puis rebooter le PC.

Organisation des fichiers résultants



3.1 Développement de l'application sous DOS

L'application doit être écrite en langage C ou dans un langage acceptant des routines écrites en C Microsoft.

L'utilisateur peut utiliser le Microsoft Visual C++.

Chaque service proposé correspond à une "fonction" au sens langage C , son utilisation se résume à un simple appel avec passage d'arguments.

C'est le programme utilisateur qui organise la gestion et la cohérence des différents services qu'il demande.

3.2 Compilation et édition de lien

Tout programme C utilisant la librairie TPUNI-TE/DOS doit impérativement terminer le paragraphe des "include" par :

 # include <UNITE.h>

L'application doit être compilée en modèle LARGE.

3.3 Mise au point et exploitation

Tester la liaison vers le ou les équipements SERVEURS en ne validant qu'une partie très simple de l'application :

- utilisation de la fonction *UNITEMirror*

ou

- utilisation de l'exemple décrit au chapitre suivant.

3.4 Exemple d'application sous DOS

L'application "EXEMPLE.C" livrée avec le logiciel et décrite dans ce chapitre peut être utilisée pour tester l'installation et le bon fonctionnement de la liaison vers un équipement automate TSX7 SERVEUR distant.
Il suffit pour cela de modifier l'adresse (*UNITEOpenConnection*) et de la faire correspondre à celle de l'équipement auquel on désire accéder.
Cette application étant livrée sous forme de fichier "source", elle doit être compilée avant utilisation.
Un fichier de commande de compilation/édition de lien est fourni avec l'exemple, taper la commande suivante pour obtenir un exécutable :
comp exemple pour une compilation en modèle LARGE

3.4-1 Description du programme

Le programme effectue les opérations suivantes :
- Initialisation du contexte de communication
- Ouverture d'un canal de communication avec le coprocesseur
- Lecture des mots W10 à W50 par la fonction UNITEReadObject
- Lecture des mots W60 à W260 par la fonction UNITEReadWordArray
- Fermeture du canal de communication
- Libération du contexte de communication.

3.4-2 Programme source "Exemple.C".

```
/                                                                        /
*************************************************************************
//
//          Exemple d'utilisation de la librairie PC-DOS
//
//
/                                                                        /
*************************************************************************

/*  Objectifs :

          - exemples de lecture de variables
    Historique :

          Création le 19 Janvier 1994
    Algorithme du test :
          - Initialisation du driver
          - Ouverture d'un canal de communication
          - Lecture des mot internes 10 à 50 dans la station
          - Lecture des mot internes 60 à 260 dans la station
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unite.h>   //Prototype des fonction UNITE + constantes

void main(void)
{
//  VARIABLES INTERNES
    HEQUIP hEquip;
    HREQ hrequete;
    UNITE_RC rc;
    short buffer1[41],buffer2[201];
    USHORT i;
    DEVICEADD  Device;

// INITIALISATION DES ADRESSES
Device.uchBoard   = 1;
Device.uchNetwork  = 0;
Device.uchStation  = 254;
Device.uchGate    = 0;
Device.uchModule   = 0;
Device.uchDevice   = 0;
```

```
// OUVERTURE DU DRIVER
rc = UNITEInitDriver(1);
if (rc < 0 ) {
      printf("\nPROBLEME INITIALISATION status= %d ",rc);
      exit(0);
      }


// OUVERTURE DU CANAL DE COMMUNICATION AVEC LA STATION
hEquip = UNITEOpenConnection(&Device);
if (hEquip<0) {
      printf("\nPROBLEME OPEN CONNEXION status= %d ",hEquip);
      if (hEquip == ENOK)
      printf("\n usExtinfo = %d ",usExtinfo);
      exit(0);
      }

// Lecture des mot internes 10 à 50 dans la station
hrequete = UNITEReadObject(hEquip,0x68,0x07,10,41,(PCHAR)buffer1);
if(hrequete<0) {
      printf("\n PB REQUEST Read Object hrequete =%d  ",hrequete);
      if (hrequete == ENOK)
      printf("\n usExtinfo = %d ",usExtinfo);
      exit(0);
      }

//    REPONSE LECTURE
rc=UNITEResponse(hrequete,WAIT);
if(rc<0) {
      printf("\n PB RESPONSE Read Object rc =%d",rc);
      if (rc == ENOK)
      printf("\n usExtinfo = %d ",usExtinfo);
      exit(0);
      }

//    Affichage des valeurs lues
for (i=0;i<41;i++) {
      printf("\n  valeur[%d] = %d",i,buffer1[i]);
      }
```

```
// Lecture des mot internes 60 à 260 dans la station
hrequete=UNITEReadWordArray(hEquip,60,201,buffer2);
if(hrequete<0) {
     printf("\n PB REQUEST Read Word Array hrequete[0] =%d
     ",hrequete);
if (hrequete == ENOK)
     printf("\n usExtinfo = %d ",usExtinfo);exit(0);
     }

//   Affichage des valeurs lues
for (i=0;i<201;i++) {
     printf("\n  valeur[%d] = %d",i,buffer2[i]);
     }


//  FERMETURE DU CANAL DE COMMUNICATION
rc = UNITECloseConnection(hEquip);
if (rc != OK) {
     printf("\nErreur UNITECloseConnection n¯%d : %d",i,rc);
if (rc == ENOK)
     printf("\n usExtinfo = %d ",usExtinfo);
     }

rc = UNITECloseDriver(1);
if (rc != OK) {
     printf("\nErreur UNITECloseDriver : %d",rc);
if (rc == ENOK)
     printf("\n usExtinfo = %d ",usExtinfo);
     }
}
```

4.1 Développement de l'application sous OS/2

L'application doit être écrite en langage C ou dans un langage acceptant des routines écrites en C Microsoft.

L'utilisateur peut utiliser :

. le C Microsoft Version 6.0 (16-Bit)
. l'IBM C/2 Version 1.3 (16-Bit)
. l'IBM C Set Version 2.0 (32-Bit)

Le logiciel est découpé selon un modèle client-serveur et laisse libre la gestion multitâche d'OS/2. Ainsi, plusieurs applications utilisateurs peuvent faire appel aux services de manière asynchrone.

Chaque service proposé correspond à une "fonction" au sens langage C , son utilisation se résume à un simple appel avec passage d'arguments.

C'est le programme utilisateur qui organise la gestion et la cohérence des différents services qu'il demande.

Le lancement d'une application doit être précédé du lancement du serveur TP UNI-TE OS/2.

4.2 Compilation et édition de lien

Tout programme C utilisant la librairie TP UNI-TE/OS/2 doit impérativement commencer par :

 # include <UNITE.h> (environnement 16-bit)

ou

 # include <UNITE32.h> (environnement 32-bit)

L'application doit être compilée en modèle LARGE (environnement 16-bit).

4.3 Installation du driver et connexion

Se reporter à la documentation du coprocesseur PL7.3

4.4 Mise au point et exploitation

Tester la liaison vers le ou les équipements SERVEURS distants en ne validant qu'une partie très simple de l'application :

- utilisation de la fonction *UNITEMirror*

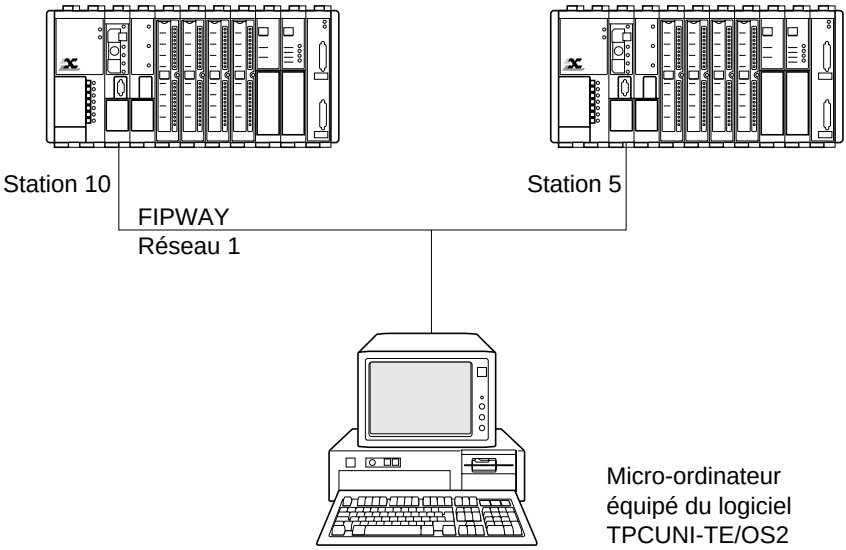
ou

- utilisation de l'exemple décrit au chapitre suivant.

4.5 Exemple d'application sous OS/2

L'application "EXEMPLE.C" livrée avec le logiciel et décrite dans ce chapitre peut être utilisée pour tester l'installation et le bon fonctionnement de la liaison vers un équipement automate TSX7 SERVEUR distant.
Il suffit pour cela de modifier l'adresse (*UNITEOpenConnection*) et de la faire correspondre à celle de l'équipement auquel on désire accéder.
Cette application étant livrée sous forme de fichier "source", elle doit être compilée avant utilisation.
Un fichier de commande est fourni avec l'exemple, taper "comp exemple" pour obtenir un exécutable (environnement de développement 16 bit).

4.5-1 Architecture correspondant à l'exemple



4.5-2 Description du programme

Le programme effectue les opérations suivantes :

- Initialisation du contexte de communication
- Ouverture d'un canal de communication
- Lecture du mot interne W50 dans l'automate
- Récupération de la réponse
- Ecriture du mot interne W100 dans l'automate
- Récupération de la réponse
- Transfert du programme TSX.APP du TSX vers le PC
- Récupération de la réponse
- Fermeture du canal de communication
- Libération du contexte de communication

4.5-3 Programme source "Exemple.C".

```
/                                                                        /
*************************************************************************
//          Exemple.C
/                                                                        /
*************************************************************************

/*  Objectifs :

          - exemples de lecture, écriture de variables et de
transfert de programme

    Historique :

          Création le 10 Décembre 1993

    Algorithme du test :
          - Initialisation du driver
          - Ouverture du canal de communication
          - Lecture du mot interne 50
          - Ecriture du mot interne 100
          - Transfert du programme du TSX -> PC */

/*FICHIERS IMPORTES */

#include <stdlib.h>
#include <stdio.h>
#include <os2.h>
#include <unite.h>


/*  VARIABLES GLOBALES */

  static DEVICEADD Equip1Adress = {1,0,1,10,0,0,0};
  static DEVICEADD Equip2Adress = {1,0,1,5,0,0,0};
  HEQUIP           hequip[10]   = {1,0,1,5,0,0,0};
  HREQ             hrequete[10];
  UNITE_RC         rc;
  short            valeur;
  USHORT           i;
void main (void)
{
```

```
/                                                                              /
*******************************************************************************
// INITIALISATION DU CONTEXTE CLIENT//
*******************************************************************************

rc = UNITEInitDriver();
if(rc<0)
      {
      printf("\nPROBLEME INITIALISATION staétus=%d.",rc); exit(0);
      }
else
      printf("\n INIT DRIVER OK.\n");

/                                                                              /
*******************************************************************************
// OUVERTURE D'UNE CONNEXION AVEC LA STATION 10//
*******************************************************************************
hEquip[1] = UNITEOpenConnection(&Equip1Adress);
if (hEquip[1]<0)
      {
     printf("\nPROBLEME OPEN CONNEXION 1 status= %d ",hequip[1]); exit(0);
      }
else
      printf ("\n 1.re CONNEXION OUVERTE hequip[1]=%d.\n",hequip[1];

/                                                                              /
*******************************************************************************
// OUVERTURE D'UNE CONNEXION AVEC LA STATION 5//
*******************************************************************************
hEquip[2] = UNITEOpenConnection(&Equip2Adress);
if (hEquip[2]<0)
      {
      printf("\nPROBLEME OPEN CONNEXION 2 status= %d ",hequip[2]);
      exit(0);
      }
else
      printf ("\n 2.me CONNEXION OUVERTE hequip[2]=%d.\n",hequip[2];
```

```
*****************************************************************************
// LECTURE DU MOT W50 DANS LA STATION 10/
*****************************************************************************
hrequete[1] = UNITEReadInternalWord(hequip[1],50, &valeur;
if(hrequete[1]<0)
      {
      printf("\n PB REQUEST Read hrequete[1] =%d.",hrequete[1]);
      exit(0);
      }
else
      printf("\n DEMANDE DE LECTURE W50 OK.\n");

*****************************************************************************
// REPONSE LECTURE W50/
*****************************************************************************
rc=UNITEResponse((HREQ)hrequete[1],WAIT);
if(rc<0)
      {
      printf("\n PB RESPONSE Read status=%d.",rc); exit(0);
      }
else
      printf("\n RESPONSE W50 OK valeur = %d.\n",valeur);

*****************************************************************************
// ECRITURE DE W100 DANS LA STATION 10/
*****************************************************************************

hrequete[1]=UNITEWriteInternalWord(hequip[1], 100, valeur);
if (hrequete[1]<0
      {
      printf("\n PB REQUEST Read hrequete[1]=%d, valeur=%d; ",
            hrequete[1],valeur); exit(0);
      }
else
      printf("\n DEMANDE D'ECRITURE W100 OK\n");

*****************************************************************************
// REPONSE ECRITURE W100/
*****************************************************************************
rc = UNITEResponse (hrequete[1],WAIT);
if (rc<0)
      {
      printf("\n PB RESPONSE Write status=%d.",rc); exit(0);
      }
else
      printf("\n Reponse W100 OK\n");
```

```
*****************************************************************************
// TRANSFERT DU PROGRAMME DE LA STATION 10/
*****************************************************************************
hrequete[1]=UNITETransferTsxPC(hequip[1],"TSX.APP");
if (hrequete[1]<0)
      {
     printf ("\n Erreur UNITETransferTsxPC:%d",hrequete[1];exit(0);
      }
else
      printf ("\n DEMANDE DE Transfert TEST.APP OK.\n");

*****************************************************************************
// REPONSE TRANSFERT/
*****************************************************************************
rc = UNITEResponse ((HREQ)hrequete[1],WAIT);
if (rc<0)
      {
      printf("\n probleme de RESPONSE avec status=%d\n",rc); exit(0);
      }
else
      printf("\n RESPONSE TRANSFERT OK.\n");

*****************************************************************************
// FERMETURE DES CONNEXIONS DE L'AUTOMATE/
*****************************************************************************
rc = UNITECloseConnection(hequete[1]);
if (rc<0)
      {
     printf("\n Erreur UNITECloseConnection status=%d\n",rc); exit(0);
      }
else
      printf("\n 1.re CONNEXION FERMEE.\n");
      printf("\n 2.eme CONNEXION FERMEE.\n");

*****************************************************************************
// CLOSE DRIVER/
*****************************************************************************
rc = UNITECloseDriver ();
if (rc<0)
      {
      printf("\n Erreur UNITECloseDriver:%d\n",rc); exit(0);
      }
else
      printf("\n CLOSE DRIVER OK.\n");
      }
```

5.1 Développement de l'application sous OS/2

«Txt mod»

L'application doit être écrite en langage C ou dans un langage acceptant des routines écrites en C Microsoft.

L'utilisateur peut utiliser le Microsoft Visual C++

Chaque service proposé correspond à une "fonction" au sens langage C , son utilisation se résume à un simple appel avec passage d'arguments.

C'est le programme utilisateur qui organise la gestion et la cohérence des différents services qu'il demande.

5.2 Compilation et édition de lien

Tout programme C utilisant la librairie TP UNI-TE/WINDOWS doit impérativement terminer le paragraphe des <include> par :

 # include <UNITE.h>

L'application peut être compilée en modèle SMALL, MEDIUM ou LARGE.

5.3 Mise au point et exploitation

Tester la liaison vers le ou les équipements SERVEURS distants en ne validant qu'une partie très simple de l'application :

- utilisation de la fonction *UNITEMirror*

ou

- utilisation de l'exemple décrit au chapitre suivant.

5.4 Exemple d'application

L'application "EXEMPLE1.C" livrée avec le logiciel et décrite dans ce chapitre peut être utilisée pour tester l'installation et le bon fonctionnement de la liaison vers un équipement automate TSX7 SERVEUR distant.
Il suffit pour cela de modifier l'adresse (*UNITEOpenConnection*) et de la faire correspondre à celle de l'équipement auquel on désire accéder.

«Txt mod»

5.4-1 Description du programme

Le programme effectue les opérations suivantes :

- Initialisation du contexte de communication
- Ouverture d'un canal de communication (avec la station 5)
- Lecture des mots internes 10 à 50 par la fonction UNITEReadObject
- Récupération de la réponse
- Lecture des mots internes 60 à 260 par la fonction UNITEReadWordArray
- Fermeture du canal de communication (avec la station 5)
- Libération du contexte de communication.

5.4-2 Programme source "Exemple1.C".

```
/                                                                        /
**************************************************************************
//          Exemple d'utilisation de la DLL XWAY WINDOWS
/                                                                        /
**************************************************************************

/*
Objectif : exemple de lecture de variables

Algorithme du test :
     - Initialisation du driver à l'ouverture de la fenêtre
     - Ouverture du canal de communication
     -  Lecture  des  mots  internes  10  à  50  avec  la  requête
     UNITEReadObject
     -  Lecture  des  mots  internes  60  à  260  avec  la  requête
     UNITEReadWordArray
     - Fermeture du canal de communication
*/

#include <windows.h>
#include <string.h>
#include "example.h"
#include "resource.h"
#include <stdio.h>
#include <unite.h>   prototype des fonctions de la DLL
                     et définition des constantes */
```

```
/*  VARIABLES INTERNES */

HANDLE     hlnst;
char       buffer[150];
int        status;
HANDLE     hLib;
HEQUIP     equip[2];

/*  INITIALISATION DE LA STRUCTURE DEVICE

Drivers :   1
Inutilise : 0
Reseau :    1
Station :   5
Porte :     0
Module :    0
Voie :      0

*/
DEVICEADD  device = {1,0,1,5,0,0,0};

HWND hEditWnd;  /* handle to edit windows */
HWND hwnd;      /* handle to main windows */

****************************************************************************
// FUNCTION: WinMain(HANDLE, HANDLE, LPSTR, int)
   PURPOSE: calls initialization function, processes message
loop//
****************************************************************************

int PASCAL WinMain(HANDLE hInstance,
                HANDLE hPrevInstance,
                LPSTR IpCmdLine,
                int nCmdShow)
{
MSG        msg;
FARPROC    IpfnInit; // pointeur sur la fonction UNITEInit
FARPROC    IpfnOpen; // pointeur sur la fonction UNITEOpenConnection

/*chargement de la DLL UPCWIN_M en modele medium */
hLib = LoadLibrary;
if (hLib<32)
     return FALSE;
```

«Txt mod»

```
// Initialisation du driver
IpfnInit = GetProcAddress(hLib, "UNITEInitDriver");
status=(*IpfnInit)(1);

// Ouverture du canal de communication
IpfnOpen = GetProcAddress(hLib, "UNITEOpenConnection");
equip[0]=(*IpfnOpen)((PDEVICEADD)&device);

if (!hPrevInstance)
if (!InitApplication(hInstance))
     return (FALSE);

if (!InitInstance(hInstance, nCmdShow))
     return (FALSE);

while (GetMessage(&msg, NULL, NULL, NULL)) {
     TranslateMessage(&msg);
     DispatchMessage(&msg);
}
return (msg.wParam);
}


/****************************************************************
     FUNCTION: MainWndProc(HWND, unsigned, WORD,LONG)
     PURPOSE: Processes messages
     MESSAGES:
     WM_COMMAND - application menu (About dialog box)
     WM_DESTROY - destroy window
*****************************************************************************/
long FAR PASCAL MainWndProc(HWND hWnd,
                unsigned message,
                WORD wParam,
                LONG lParam)
{
FARPROC         IpfnReadWordArray;
FARPROC         IpfnResponse;
FARPROC         IpfnReadObject;
FARPROC         IpfnClose;
FARPROC         IpfnReadCloseC;
static int      Req[2];
WORD            IpuBufobj[41];
WORD            IpuBufArray[201];
```

```
switch (message) {
case WM_COMMAND:
     switch (wParam) {
     /* file menu commands */

case ID_LISTEDEMOTS_START
     // Utilisation de la requete Lecture liste de mots: requete
synchrone
     IpfnReadWordArray = GetProcAddress(hLib, "UNITEReadWordArray");
     Req[0]=(*IpfnReadWordArray)     (HEQUIP)equip[0],
                                     (USHORT)60,//debut lecture
                                     (USHORT)201,//quantite
                                     (PSHORT)IpuBufArray;
     break;

case ID_REQUETEOBJET_START
     // Utilisation de la requete Lecture objet: requete
asynchrone
     IpfnReadObject = GetProcAddress(hLib, "UNITEReadObject");
     Req[1]=(*IpfnReadObject)  (HEQUIP)equip[0],
                               (UCHAR)104,
                               (UCHAR)7,
                               (USHORT)10,//debut lecture
                               (USHORT)41,//quantite
                               (PUCHAR)IpuBufobj;
if (Req[1]>=0
{
     /*recuperation de la reponse a la requete Req[1]*/
     IpfnResponse = GetProcAddress(hLib,"UNITEResponse");
     status=(*IpfnResponse)(Req[1],WAIT);
}
break;

case IDM_EXIT:
     DestroyWindow(hWnd);
     break;
}
break;
```

«Txt mod»

```
Case WM_DESTROY:
      // Fermeture du canal de communication
      IpfnCloseC=GetProcAdress(hLib, "UNITECloseConnection");
      status=(*IpfnCloseC)(equip[0]);

      /* Fermeture du driver */
      IpfnClose=GetProcAdress(hLib, "UNITECloseDriver");
      status=(*IpfnClose)(1);

      // dechargement de la DLL
      FreeLibrary(hLib);

      PostQuitMessage(0);
      break;

default:
return (DefWindowProc(hWnd, message, wParam,Iparam));
}
return (NULL);
}
```

Sommaire

6.1 Fonctions : gestion du contexte de communication

6.1-1 UNITEInitDriver

• Sous DOS et WINDOWS

Description :

Initialise le contexte de communication. Toute application client doit débuter par UNITEInitDriver.

Syntaxe :

UNITE_RC rc = UNITEInitDriver (USHORT usNb_drv);

En entrée :

USHORT usNb_drv : Nombre de drivers 1

En retour :

- soit un code OK si tout s'est bien passé,
- soit un code d'erreur négatif :

EBORNES	le paramètre en entrée **usNb_drv** est invalide.
EDRVAOPEN	le ou les drivers sont déja ouvert(s).
ENOK	il y a une erreur générale remontée par le driver. Consultez la variable globale **usExinfo** pour en connaître la cause (cf: Tables des erreurs).

• Sous OS/2

Description :

Initialise le contexte de communication. Toute application client doit débuter par UNITEInitDriver.

Syntaxe :

UNITE_RC rc = UNITEInitDriver ();

En retour :

- soit un code OK si tout s'est bien passé,
- soit un code d'erreur négatif :

NORESAIVAIBLE	plus de ressource disponible.
MPV_INITDONE	init client déjà réalisée.

6.1-2 UNITECloseDriver

• Sous DOS et WINDOWS

Description :

Libère le contexte de communication. Terminer l'application client par cette fonction.

Syntaxe :

UNITE_RC rc = UNITECloseDriver(USHORT usNb_drv);

En entrée :

USHORT usNb_drv : Nombre de drivers à fermer = 1.

En retour :

- soit un code OK si tout s'est bien passé,
- soit un code d'erreur négatif :

EBORNES	le paramètre en entrée **usNb_drv** est invalide.
EDRVAOPEN	le ou les drivers n'ont pas été ouverts.
ENOK	il y a une erreur générale remontée par le driver. Consultez la variable globale **usExinfo** pour en connaître la cause (cf: Tables des erreurs).

• Sous OS/2

Description :

Libère le contexte de communication. Terminer l'application client par cette fonction.

Syntaxe :

UNITE_RC rc = UNITECloseDriver();

En retour :

- soit un code OK si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINIT	driver non initialisé.

6.1-3 UNITEOpenConnection

Description :

Alloue un canal de communication du driver vers un équipement distant.

Syntaxe :

HEQUIP Hequip = UNITEOpenConnection(PDEVICEADD pDeviceAdd);

En entrée :

PDEVICEADD pDeviceAdd : adresse de l'équipement distant

avec typedef struct {

UCHAR	uchBoard;	:	numéro de driver
UCHAR	uchMode;	:	non utilisé, mettre à 0
UCHAR	uchNetwork;	:	\ correspondent à
UCHAR	uchStation;	:	| l'adresse destinataire
UCHAR	uchGate;	:	| XWAY, telle qu'elle est
UCHAR	uchModule;	:	| définie dans le protocole
UCHAR	uchDevice;	:	/ UNI-TE.

} DEVICEADD, * PDEVICEADD;

En retour DOS et WINDOWS :

soit un code (>=0) si tout s'est bien passé
Ce code nommé hEquip identifie un équipement.
Il sera passé en argument à chaque appel fonction qui est destiné à cet équipement.
soit un code d'erreur négatif :

EBORNES	Le paramètre en entrée **uchBoard** est invalide.
EDRVAOPEN	Le ou les drivers sont déjà ouverts.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause.(cf:Tables des erreurs)
ENOMORESOCK	Toutes les voies de communication disponibles ont été ouvertes.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
ESRCEADR	L'adresse source n'est pas valide.
EDSTADR	L'adresse de l'équipement distant n'est pas valide.

En retour OS/2 :

soit un code (>=0) si tout s'est bien passé
Ce code nommé hEquip identifie un équipement. Il sera passé en argument à chaque appel fonction qui est destiné à cet équipement.
soit un code d'erreur négatif :

NORESAVAILABLE	Plus de ressource disponible.
PBINITSESSION	Contexte de communication non initialisé.
MPW_NO_ok	Problème liaison.

6.1-4 UNITECloseConnection

Description :

Libère un canal de communication du driver vers un équipement.

Syntaxe :

UNITE_RC rc = UNITECloseConnection (HEQUIP hEquip);

En entrée :

hEquip : identificateur de l'équipement lié au canal à libérer.

En retour DOS et WINDOWS :

- soit OK si tout s'est bien passé,
- soit un code d'erreur négatif :

EBORNES	Le paramètre en entrée **hEquip** est supérieur au nombre maximum de connexions disponibles.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause.(cf:Tables des erreurs)
ENOTOPENSOCK	Le canal de communication spécifié par le paramètre **hEquip** n'a pas été ouvert.

En retour OS/2 :

- soit OK si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé

6.1-5 UNITEOpenUnsollicitedData

Description :

Les applications du coprocesseur ou les équipements reliés au coprocesseur par FIPWAY/FIPIO adresseront leurs données non sollicitées vers le canal de communication ouvert en réception (adresse réseau et station égale à celle du coprocesseur, numéro de porte destination égale à 3). Par bloc TXT ; bloc TXT LOCAL ou NET (LOCAL pour une application du coprocesseur, NET sur FIPWAY), type TER, le paramètre Txti,T sera initialisé à 3 dans le programme application avant son émission.

Syntaxe :

HEQUIP hEquip = UNITEOpenUnsollicitedData(UCHAR uchNumDriver);

En entrée :

uchNumDriver : numéro de driver : 1 pour DOS et WINDOWS, 0 pour OS/2

En retour DOS et WINDOWS :

- soit un numéro (>=0) identifiant le canal de communication en réception de DNS si tout s'est bien passé.
  Ce numéro hEquip sera passé en argument à chaque lecture de données non sollicitées.
- soit un code d'erreur négatif.

EBORNES	Le paramètre en entrée **uchNum_drv** est invalide.
EDRVNOTOPEN	Le driver n'a pas été ouvert.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause.(cf:Tables des erreurs)
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EUNSOOPEN	Le canal de communication alloué aux données non sollicitées a déjà été ouvert.

En retour OS/2 :

- soit un numéro (>=0) identifiant le canal de communication en réception de DNS si tout s'est bien passé.
  Ce numéro hEquip sera passé en argument à chaque lecture de données non sollicitées.
- soit un code d'erreur négatif.

MPW_OPENED	Canal données non sollicitées déjà ouvert.
PBINIT	Driver non initialisé.
NORESAVAILABLE	Plus de ressource disponible.

6.1-6 UNITECloseUnsollicitedData

Description :

Libère le canal de communication pour réception des données non sollicitées.

Syntaxe :

UNITE_RC rc = UNITECloseUnsollicitedData(UCHAR uchNumDriver);

En entrée :

uchNumDriver : numéro de driver : 1 pour DOS et WINDOWS, 0 pour OS/2.

En retour DOS et WINDOWS :

- soit un code OK si tout s'est bien passé,
- soit un code d'erreur négatif :

EBORNES	Le paramètre en entrée **uchNum_drv** est invalide.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)
ENOTOPENSOCK	Le canal de communication spécifié par le paramètre **hEquip** n'a pas été ouvert.

En retour OS/2 :

- soit un code OK si tout s'est bien passé,
- soit un code d'erreur négatif :

NORESAVAILABLE	Plus de ressource disponible
PBINITSESSION	Contexte de communication non initialisé.
PBIN	Driver non initialisé.
MPW_NO_OK	Problème liaison.

6.1-7 UNITETimeOut

Description :

Réglage du temps d'attente maximal d'exécution des requêtes.
Lorsque le logiciel TPUNI-TE émet une requête à destination d'un équipement distant, il lance une temporisation égale à ce temps.
Si dans le programme application, on active la fonction ***UNITEResponse*** et que cette temporisation est arrivée à terme sans qu'on ait obtenu de réponse, UNITEResponse renverra un code d'erreur TIME_OUT.

Syntaxe :

UNITE_RC rc = UNITETimeOut(ULONG ulTimeOut);

En entrée :

ulTimeOut : temps en millisecondes.

En retour DOS et WINDOWS :

- soit un code OK si tout s'est bien passé,
- soit un code d'erreur négatif :

EBORNES	le paramètre ulTimeOut est inférieur à -1

Remarque
A l'expiration, la transaction est abandonnée.

En retour OS/2 :

- soit un code OK si tout s'est bien passé,
- soit un code d'erreur négatif :

MPW_NO_OK	problème liaison

Remarque
A l'expiration, la transaction est abandonnée.

6.1-8 UNITEGetSystemError (OS/2 uniquement)

Description :

Récupération d'une erreur système lorsqu'une fonction retourne le code MPW_ERR_SYSTEM.

Syntaxe :

USHORT rc = UNITEGetSystemError();

En retour :

Le dernier numéro d'erreur système.

6.1-9 UNITEGetUsExtInfo (WINDOWS uniquement)

Description :

Permet de récupérer la variable UsExtInfo pour connaître le détail d'une erreur (ENOK) retournée par une fonction.

Syntaxe :

UNITE-RC rc = UNITEGetUsExtInfo (void);

En entrée :

Rien.

En retour :

La variable UsExtInfo : les codes d'erreurs liés à cette variable sont présentés en Annexe.

6.2 Fonctions : gestion des équipements

6.2-1 UNITEMirror

Description :

Permet de tester la connexion avec un équipement.
Tous les équipements serveurs UNI-TE supportent cette requête et renvoient les mêmes données que celles reçues.

Syntaxe :

```
HREQ    hReq = UNITEMirror (HEQUIP    hEquip,
                            UCHAR     uchQty);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
uchQty : nombre d'octets envoyés (de 1 à 126).

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBORNES	Le paramètre en entrée **ucQty** est supérieur à 126.
EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible
EBUILD	Erreur à la construction du datagramme
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.2-2 UNITEIdentification

Description :

Identification de l'équipement distant.
Tous les équipements serveurs UNI-TE supportent cette requête.

Syntaxe :

```
HREQ        hReq = UNITEIdentification (HEQUIP      hEquip,
                                        PUCHAR      pBuffer_Ident);
```

En entrée :

hEquip : identificateur du destinataire de la requête.

En sortie :

pBuffer_Ident : pointeur du buffer de stockage de la réponse Identification.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.2-3 UNITEReserve

Description :

Réservation de l'équipement serveur avec lequel on veut dialoguer.
La réservation démarre avec la fonction *UNITEReserve* et se termine avec la fonction *UNITERelease*.
La réservation est interrompue automatiquement au bout de 60s. si elle n'a pas été entretenue avec la fonction *UNITEIAmAlive*.
L'entretien de réservation est réalisé par n'importe quelle requête UNI-TE.
En cas d'absence de communication, la fonction *UNITEIAmAlive* permet cet entretien.

Syntaxe :

HREQ hReq = UNITEReserve (HEQUIP hEquip);

En entrée :

hEquip : identificateur du destinataire de la requête.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

Remarque

Les services critiques qui exigent une réservation (RUN, STOP, Transfert de fichier...), l'entretien de réservation et la déréservation doivent utiliser le même canal de communication que celui mis en oeuvre lors de la réservation.

6.2-4 UNITERelease

Description :

Déréservation d'un équipement.

Syntaxe :

HREQ hReq = UNITERelease(HEQUIP hEquip);

En entrée :

hEquip : identificateur du destinataire de la requête.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

Remarque

Les services critiques qui exigent une réservation (RUN, STOP, Transfert de fichier...), l'entretien de réservation et la déréservation doivent utiliser le même canal de communication que celui mis en oeuvre lors de la réservation.

6.2-5 UNITEIAmAlive

Description :

Demande d'entretien de la réservation d'un équipement. Cette réservation doit être entretenue périodiquement (environ toutes les 60 secondes) faute de quoi, l'équipement est automatiquement déréservé.

Syntaxe :

HREQ hReq = UNITEIAmAlive(HEQUIP hEquip);

En entrée :

hEquip : identificateur du destinataire de la requête.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

Remarque

Les services critiques qui exigent une réservation (RUN, STOP, Transfert de fichier...), l'entretien de réservation et la déréservation doivent utiliser le même canal de communication que celui mis en oeuvre lors de la réservation.

6.2-6 UNITERun

Description :

Mise en "Run" d'un équipement.

Remarque
Nécessite la réservation préalable de l'automate TSX7 par le même canal de communication.

Syntaxe :

HREQ hReq = UNITERun (HEQUIP hEquip);

En entrée :

hEquip : identificateur du destinataire de la requête.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.2-7 UNITEStop

Description :

Mise en "Stop" d'un équipement.

Remarque

Nécessite la réservation préalable de l'automate TSX7 par le même canal de communication.

Syntaxe :

HREQ hReq = UNITEStop(HEQUIP hEquip);

En entrée :

hEquip : identificateur du destinataire de la requête.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3 Fonctions : accès aux variables

6.3-1 UNITEReadInternalBit

Description :

Lecture d'un bit interne Bi.

Syntaxe :

```
HREQ    hReq = UNITEReadInternalBit (HEQUIP    hEquip,
                                     USHORT    usNbit,
                                     PBITVAL   pValue);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
usNbit : numéro du bit interne à lire.

En sortie :

pValue : pointeur de la structure de réponse
avec typedef struct {

```
        BYTE    bValue;      : valeur du bit lu
        BYTE    bOverride;   : état de forçage du bit lu
        }BITVAL,  * PBITVAL;
```

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-2 UNITEReadSystemBit

Description :

Lecture d'un bit bit système SYi.

Syntaxe :

```
HREQ    hReq = UNITEReadSystemBit (HEQUIP    hEquip,
                                   USHORT    usNbit,
                                   PBITVAL   pValue);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
usNbit : numéro du bit système à lire.

En sortie :

pValue : pointeur de la structure de réponse
avec typedef struct {

```
BYTE    bValue;
BYTE    bOverride;
}BITVAL,   * PBITVAL;
```

Seul le champ bValue (valeur du bit lu) est significatif.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-3 UNITEWriteInternalBit

Description :

Ecriture d'un bit interne Bi.

Syntaxe :

```
HREQ    hReq  =  UNITEWriteInternalBit ( HEQUIP    hEquip
                                         USHORT   usNbit,
                                         BOOL     boValue);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
usNbit : numéro du bit interne à écrire.
boValue : valeur à écrire (0 ou 1)

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-4 UNITEWriteSystemBit

Description :

Ecriture d'un bit système SYi.

Syntaxe :

```
HREQ    hReq  =  UNITEWriteSystemBit( HEQUIP    hEquip
                                      USHORT   usNbit,
                                      BOOL     boValue);
```

En entrée :

hEquip	: identificateur du destinataire de la requête.
usNbit	: numéro du bit système à écrire.
boValue	: valeur à écrire (0 ou 1)

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-5 UNITEReadInternalWord

Description :

Lecture d'un mot interne Wi.

Syntaxe :

```
HREQ    hReq = UNITEReadInternalWord (HEQUIP   hEquip,
                                      USHORT   usNword,
                                      PSHORT   pValue);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
usNword : numéro du mot interne à lire.

En sortie :

pValue : pointeur du mot lu.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-6 UNITEReadConstantWord

Description :

Lecture d'un mot constant CWi.

Syntaxe :

```
HREQ    hReq = UNITEReadConstantWord ( HEQUIP   hEquip,
                                       USHORT   usNword,
                                       PSHORT   pValue);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
usNword : numéro du mot constant à lire.

En sortie :

pValue : pointeur du mot lu.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-7 UNITEReadSystemWord

Description :

Lecture d'un mot système SWi.

Syntaxe :

```
HREQ    hReq = UNITEReadSystemWord ( HEQUIP    hEquip,
                                     USHORT    usNword,
                                     PSHORT    pValue);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
usNword : numéro du mot système à lire.

En sortie :

pValue : pointeur du mot lu.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-8 UNITEWriteInternalWord

Description :

Ecriture d'un mot interne Wi.

Syntaxe :

```
HREQ    hReq  = UNITEWriteInternalWord( HEQUIP   hEquip,
                                       USHORT   usNword,
                                       SHORT    sValue);
```

En entrée :

hEquip	: identificateur du destinataire de la requête.
usNword	: numéro du mot interne à écrire.
sValue	: valeur à écrire (sur 16 bits)

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-9 UNITEWriteSystemWord

Description :

Ecriture d'un mot système SWi.

Syntaxe :

```
HREQ    hReq  = UNITEWriteSystemWord( HEQUIP   hEquip,
                                     USHORT   usNword,
                                     SHORT    sValue);
```

En entrée :

hEquip	: identificateur du destinataire de la requête.
usNword	: numéro du mot système à écrire.
sValue	: valeur à écrire (sur 16 bits)

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-10 UNITEReadCommonWord

Description :

Lecture d'un mot commun COMi,j.
Avec i = numéro de station, et j = numéro du mot.

Syntaxe :

```
HREQ hReq  = UNITEReadCommonWord (HEQUIP        hEquip,
                                  UCHAR         uchStation,
                                  USHORT        usNWord,
                                  PCOMWORDVAL   pValue);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
uchStation : numéro de station.
usNWord : numéro de mot à lire.

En sortie :

pValue : pointeur de la structure de réponse
avec typedef struct {

```
USHORT      usNbWord;     : nombre de mots COM
                            dans la station
SHORT       sValue;       : valeur lue
}COMWORDVAL,   * PCOMWORDVAL;
```

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-11 UNITEWriteCommonWord

Description :

Ecriture d'un mot commun COMi,j .
Avec i = numéro de station, et j = numéro du mot.

Syntaxe :

```
HREQ  hReq = UNITEWriteCommonWord ( HEQUIP    hEquip,
                                    UCHAR     uchStation,
                                    USHORT    usNWord,
                                    SHORT     sValue);
```

En entrée :

hEquip	: identificateur du destinataire de la requête.
uchStation	: numéro de station du destinataire.
usNWord	: numéro du mot à écrire.
sValue	: valeur à écrire.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-12 UNITEReadInternalDWord

Description :

Lecture d'un double mot interne DWi.

Syntaxe :

```
HREQ    hReq = UNITEReadInternalDWord ( HEQUIP    hEquip,
                                        USHORT    usNDword,
                                        PLONG     pValue);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
usNDword : numéro du double mot interne à lire.

En sortie :

pValue : pointeur du double mot lu.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-13 UNITEReadConstantDWord

Description :

Lecture d'un double mot constant DCWi.

Syntaxe :

```
HREQ    hReq = UNITEReadConstantDWord (HEQUIP    hEquip,
                                       USHORT    usNDword,
                                       PLONG     pValue);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
usNDword : numéro du double mot constant à lire.

En sortie :

pValue : pointeur du double mot lu.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-14 UNITEWriteInternalDWord

Description :

Ecriture d'un double mot interne DWi.

Syntaxe :

```
HREQ hReq  = UNITEWriteInternalDWord ( HEQUIP    hEquip,
                                       USHORT    usNDword,
                                       LONG      lValue);
```

En entrée :

hEquip	: identificateur du destinataire de la requête.
usNDword	: numéro du double mot à écrire.
lValue	: valeur à écrire (sur 32 bits)

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-15 UNITEReadObject

Description :

Lecture d'une suite d'objets.

Remarque

La longueur maximale des données (nombre d'objets multiplié par la taille d'un objet en octets) dépend du serveur UNI-TE interrogé.
Elle est de 120 octets pour un automate TSX ou une commande numérique NUM, et de 30 octets pour les TSX 17.

Syntaxe :

```
HREQ   hReq  =  UNITEReadObject (HEQUIP     hEquip,
                                 UCHAR      uchSegment,
                                 UCHAR      uchType,
                                 USHORT     usFirst,
                                 USHORT     usQty,
                                 PCHAR      pValue);
```

En entrée :

hEquip	: identificateur du destinataire de la requête.
uchSegment	: spécifie le numéro de segment. (Voir Chap. 6.4)
uchType	: spécifie le type de l'objet (double mot, mot, octet, ...) (Voir Chap. 6.4)
usFirst	: adresse du premier objet à lire.
usQty	: nombre d'objets consécutifs à lire.

En sortie :

pValue	: adresse du buffer où sont stockées les données lues.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-16 UNITEWriteObject

Description :

Ecriture d'une suite d'objets.

Remarque

La longueur maximale des données (nombre d'objets multiplié par la taille d'un objet en octets) dépend du serveur UNI-TE interrogé.
Elle est de 120 octets pour un automate TSX ou une commande numérique NUM, et de 30 octets pour les TSX 17.

Syntaxe :

```
HREQ    hReq = WriteObject (HEQUIP      hEquip,
                            UCHAR       uchSegment,
                            UCHAR       uchType,
                            USHORT      usFirst,
                            USHORT      usQty,
                            UCHAR       uchSize,
                            PUCHAR      pValue);
```

En entrée :

hEquip	: identificateur du destinataire de la requête.
uchSegment	: spécifie le numéro de segment. (Voir Chap. 6.4)
uchType	: spécifie le type de l'objet (double mot, mot, octet, ...) (Voir Chap. 6.4)
usFirst	: adresse du premier objet à écrire.
usQty	: nombre d'objets consécutifs à écrire.
usSize	: taille de la suite d'objets à écrire (en octets)
pValue	: adresse du buffer où sont stockées les données à écrire.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFUL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-17 UNITEReadWordArray

Description :

Lecture d'une table de mots internes (segment 104, octet spécifique 7).

Remarque

La longueur maximale de la table est de 16 kMots.

Syntaxe :

```
HREQ    hReq   = UNITEReadWordArray (HEQUIP    hEquip,
                                      USHORT   usFirst,
                                      USHORT   usQty,
                                      PSHORT   pValue);
```

En entrée :

hEquip	: identificateur du destinataire de la requête.
usFirst	: adresse du premier mot de la table.
usQty	: nombre de mots à lire (inférieur à 16000).

En sortie :

pValue	: adresse du buffer contenant les mots de la table lue.

En retour DOS et WINDOWS :

- soit OK si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

Remarque

Cette fonction renvoie son code retour après exécution complète. Elle ne nécessite donc pas de demande de réponse UNITEResponse. Le canal associé reste bloqué pendant ce temps.

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.3-18 UNITEWriteWordArray

Description :

Ecriture d'une table de mots internes (segment 104, octet spécifique 7).

Remarque

La longueur maximale de la table est de 16 kMots.

Syntaxe :

```
HREQ    hReq =  UNITEWriteWordArray ( HEQUIP    hEquip,
                                      USHORT    usFirst,
                                      USHORT    usQty,
                                      PSHORT    pValue);
```

En entrée :

hEquip	: identificateur du destinataire de la requête.
usFirst	: adresse du premier mot de la table.
usQty	: nombre de mots à écrire (inférieur à 16000).
pValue	: adresse du buffer des valeurs à écrire.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

Remarque

Cette fonction renvoie son code retour après exécution complète. Elle ne nécessite donc pas de demande de réponse UNITEResponse. Le canal associé reste bloqué pendant ce temps.

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé.
MPW_CLIENTNOK	Canal de communication non alloué.
MPW_REQPENDING	Requête en cours sur le canal de communication.
MPW_SEQ	Problème enchaînement sur serveur.
NORESAVAIBLE	Pas ou plus de ressource disponible.
MPW_REQABORTED	Requête abandonnée.
PBDRV	Problème driver Prise Terminal.
RESPREF	Code réponse refusée.
MPW_ERR_SYSTEM	Erreur système.

6.4 Fonctions : gestion de domaine

6.4-1 UNITETransferTsxPC

Description :

Déchargement de programme TSX série 40 vers PC.

Syntaxe :

```
HREQ    hReq  =  UNITETransferTsxPC (HEQUIP      hEquip,
                                     PCHAR      pchFileName);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
pchFileName : Nom du fichier application avec extension.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)
EFICH	Erreur à l'ouverture du fichier.
ELECFICH	Erreur à la lecture du fichier.
ENOTSXV4	La version du processeur du TSX est inférieure à V6.
ETYPTSXDIFF	Le type de TSX et celui décrit dans le fichier sont différents.
ECOMPATINOK	Le fichier et le TSX ne sont pas compatibles.
EROMCART	La cartouche du TSX est une ROM.
ENOVALIDCART	La cartouche du TSX n'est pas valide.
EAPIOUTMEM	La taille du fichier dépasse celle de la cartouche.
ETSXETATNOK	Le TSX n'est pas en état d'opérer des chargement ou déchargements.
EDATAORDER	Erreur de cohérence de données.
ESEQ	Erreur de séquencement.
ETSXWRITE	Ecriture impossible.
EVERIFAPPNOK	Application invalide.

Remarque

Cette fonction renvoie son code retour après exécution complète. Elle ne nécessite donc pas de demande de réponse UNITEResponse. Le canal associé reste bloqué pendant ce temps.

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

ERRFICH	Erreur de fichier
NOFICHAPP	Le fichier n'est pas un fichier Application
ERRECRFICH	Erreur écriture fichier
ERRLECFICH	Erreur lecture fichier
NAKRESERV	Réservation TSX impossible
NOTSXV4	Le TSX distant n'est pas de type V4
TYPTSXDIFF	Les TSX fichier et distant sont différents
TSXETATNOK	L'état du TSX est incompatible avec le chargement
COMPATINOK	Problème de compatibilité TSX
APIPROT	Application protégée
NOAPI	Application inexistante
ROMCART	La cartouche est une ROM
NOVALIDCART	Cartouche non valide
APIOUTMEM	Application trop importante pour TSX distant
NAKSTOP	Impossible de stopper le TSX distant
ERRSEQ	Erreur de séquencement
ERRTSXWRITE	Ecriture du TSX impossible
ERRDATAORDER	Données incohérentes
NAKENTRESERV	Erreur entretien réservation
ENDSEQNAK	Fin de séquence de déchargement refusée
VERIFAPPNOK	Erreur après vérification Appli non valide
PBINITSESSION	Contexte de communication non initialisé
PBSHUTDOWN	Problème de fermeture
NORESAVAIBLE	Problème driver Prise Terminal
NOK	Pas encore de réponse
PBRECEPTRAME	Erreur sur réception
NACK	Message reçu mais non traité
RESPREF	Code réponse refusé reçu
PBINITSESSION	Contexte de communication non initialisé
MPW_CLIENTNOK	Canal de communication non alloué
MPW_REQPENDING	Requête en cours sur le canal de communication
MPW_SEQ	Problème enchaînement sur serveur
NORESAVAIBLE	Pas ou plus de ressource disponible
MPW_REQABORTED	Requête abandonnée
PBDRV	Problème driver Prise Terminal
RESPREF	Code réponse refusé
MPW_ERR_SYSTEM	Erreur système

6.4-2 UNITETransferPCTsx

Description :

Chargement de programme TSX série 40 à partir du PC.

Syntaxe :

```
HREQ    hReq  =  UNITETransferPcTsx (HEQUIP      hEquip,
                                    PCHAR       pchFileName);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
pchFileName : Nom du fichier application avec extension.

En retour DOS et WINDOWS :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)
EFICH	Erreur à l'ouverture du fichier.
ELECFICH	Erreur à la lecture du fichier.
ENOTSXV4	La version du processeur du TSX est inférieure à V6.
ETYPTSXDIFF	Le type de TSX et celui décrit dans le fichier sont différents.
ECOMPATINOK	Le fichier et le TSX ne sont pas compatibles.
EROMCART	La cartouche du TSX est une ROM
ENOVALIDCART	La cartouche du TSX n'est pas valide.
EAPIOUTMEM	La taille du fichier dépasse celle de la cartouche.
ETSXETATNOK	Le TSX n'est pas en état d'opérer des chargement ou déchargements.
EDATAORDER	Erreur de cohérence de données.
ESEQ	Erreur de séquencement.
ETSXWRITE	Ecriture impossible.
EVERIFAPPNOK	Application invalide.

Remarque

Cette fonction renvoie son code retour après exécution complète. Elle ne nécessite donc pas de demande de réponse UNITEResponse. Le canal associé reste bloqué pendant ce temps.

En retour OS/2 :

- soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

ERRFICH	Erreur de fichier
NOFICHAPP	Le fichier n'est pas un fichier Application
ERRECRFICH	Erreur écriture fichier
ERRLECFICH	Erreur lecture fichier
NAKRESERV	Réservation TSX impossible
NOTSXV4	Le TSX distant n'est pas de type V4
TYPTSXDIFF	Les TSX fichier et distant sont différents
TSXETATNOK	L'état du TSX est incompatible avec le chargement
COMPATINOK	Problème de compatibilité TSX
APIPROT	Application protégée
NOAPI	Application inexistante
ROMCART	La cartouche est une ROM
NOVALIDCART	Cartouche non valide
APIOUTMEM	Application trop importante pour TSX distant
NAKSTOP	Impossible de stopper le TSX distant
ERRSEQ	Erreur de séquencement
ERRTSXWRITE	Ecriture du TSX impossible
ERRDATAORDER	Données incohérentes
NAKENTRESERV	Erreur entretien réservation
ENDSEQNAK	Fin de séquence de déchargement refusée
VERIFAPPNOK	Erreur après vérification Appli non valide
PBINITSESSION	Contexte de communication non initialisé
PBSHUTDOWN	Problème de fermeture
NORESAVAIBLE	Problème driver Prise Terminal
NOK	Pas encore de réponse
PBRECEPTRAME	Erreur sur réception
NACK	Message reçu mais non traité
RESPREF	Code réponse refusé reçu
PBINITSESSION	Contexte de communication non initialisé
MPW_CLIENTNOK	Canal de communication non alloué
MPW_REQPENDING	Requête en cours sur le canal de communication
MPW_SEQ	Problème enchaînement sur serveur
NORESAVAIBLE	Pas ou plus de ressource disponible
MPW_REQABORTED	Requête abandonnée
PBDRV	Problème driver Prise Terminal
RESPREF	Code réponse refusé
MPW_ERR_SYSTEM	Erreur système

Remarque
Sous OS/2, cette fonction nécessite une demande de réponse UNITEResponse, le tranfert doit être lancé avec l'automate TSX modèle 40 en Stop.

6.5 Fonctions : données non sollicitées

6.5-1 UNITEUnsollicitedData DOS et WINDOWS

Description :

Réception de données non sollicitées. (Se référer à UNITEOpenUnsollicitedData).

Syntaxe :

```
UNITE_RC rc = UNITEUnsollicitedData ( HEQUIP hEquip,
                                      PREADUNSOLLICITEDDATA pValue);
```

En entrée :

hEquip : identificateur du canal de communication ouvert pour récupérer les données non sollicitées.

En sortie :

```
pValue                : buffer de récupération des données.
avec typedef struct   {
                      UCHAR     uchBoard
                      UCHAR     uchNetwork;  :  \
                      UCHAR     uchStation;  :  | adresse émettrice
                      UCHAR     uchGate;     :  | définie dans le
                      UCHAR     uchModule;   :  | protocole XWAY.
                      UCHAR     uchDevice;   :  /
                      UCHAR     uchQty       : nombre d'octets utiles
                      CHAR      chData[LG_MAX_UNSOL];
                      } READUNSOLLICITEDDATA, *PREADUNSOLLICITEDDATA;
#define LG_MAX_UNSOL      126
```

En retour DOS et WINDOWS :

- OK : des donnée non sollicitées sont disponibles dans le buffer de la requête.
- sinon, un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs)
PENDING	Absence de données non sollicitées.

6.5-2 UNITEUnsollicitedData OS/2

Description :

Réception de données non sollicitées. (Se référer à UNITEOpenUnsollicitedData).

Syntaxe :

```
UNITE_RC rc = UNITEUnsollicitedData ( UCHAR uchNumDriver,
                                      PREADUNSOLLICITEDDATA pValue);
```

En entrée :

uchNumDriver : numéro de driver (0 pour OS/2).

En sortie :

```
pValue                : buffer de récupération des données.
avec typedef struct   {
                      UCHAR     uchBoard
                      UCHAR     uchNetwork;  :  \
                      UCHAR     uchStation;  :  | adresse émettrice
                      UCHAR     uchGate;     :  | définie dans le
                      UCHAR     uchModule;   :  | protocole XWAY.
                      UCHAR     uchDevice;   :  /
                      UCHAR     uchQty       : nombre d'octets utiles
                      CHAR      chData[LG_MAX_UNSOL];
                      } READUNSOLLICITEDDATA, *PREADUNSOLLICITEDDATA;
#define LG_MAX_UNSOL      126
```

En retour OS/2:

- OK : des données non sollicitées sont disponibles dans le buffer de la requête.
- WAIT : pas de données non sollicitées
- sinon, un code d'erreur négatif :

PBINITSESSION	Contexte de communication non initialisé
MPW_CLIENTNOK	Canal de communication non alloué.

6.6 Fonction : fonction générique

6.6-1 UNITERequest

Description :

Emission d'une requête générique (tout type de message).

Syntaxe :

```
HREQ    hReq  =  UNITERequest ( HEQUIP        hEquip,
                               PREQUESTIN     pIn,
                               PREQUESTOUT    pOut);
```

En entrée :

hEquip : identificateur du destinataire de la requête.
pIn : pointeur du buffer contenant la requête à émettre.

avec typedef struct {

UCHAR	uchCode;	code requête
UCHAR	uchCat;	code catégorie
UCHAR	uchQty;	nombre d'octets de la requête
CHAR	chData[LG_MAX_PAR-1];	détail des octets.

}REQUESTIN, * PREQUESTIN;

En sortie

pOut : pointeur buffer de récupération de la réponse.

avec typedef struct {

UCHAR	uchCode;	code reponse
UCHAR	uchQty;	nombre d'octets de la réponse
CHAR	chData[LG_MAX_PAR-1];	détail des octets.

} REQUESTOUT, * PREQUESTOUT;

avec LG_MAX_PAR = 126

En retour DOS et WINDOWS :

- Soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

EBADPARAM	Le paramètre en entrée **hEquip** est inférieur à 0.
ECONNOTOPEN	Le canal de communication spécifié par la valeur de **hEquip** n'est pas ouvert.
EBADBUFFER	Les buffers internes n'ont pas été alloués.
EMEMFULL	Il n'y a plus de mémoire libre disponible.
EBUILD	Erreur à la construction du datagramme.
ENOK	Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

- Soit un identificateur de requête (>=0) si tout s'est bien passé,
- soit un code d'erreur négatif :

PBINITSESSION	Contexte de communication on initialisé
MPW_CLIENTNOK	Canal de communication non alloué
MPW_REQPENDING	Requête en cours sur le canal de communication
MPW_SEQ	Problème enchaînement sur serveur
NORESAVAIBLE	Pas ou plus de ressource disponible
MPW_REQABORTED	Requête abandonnée
PBDRV	Problème driver Prise Terminal
RESPREF	Code réponse refusé
MPW_ERR_SYSTEM	Erreur système

6.7 Fonction : récupération des réponses

6.7-1 UNITEResponse

Description :

Fonction demande de réponse UNI-TE.

Syntaxe :

```
UNITE_RC rc  =  UNITEResponse (      HREQ        hReq,
                                     short       sMode);
```

En entrée :

hReq : identificateur de requête.
sMode : mode de fonctionnement.

- WAIT : réponse demandée avec attente infinie.
- NOWAIT : résultat de la requête demandée sans attente.

En retour DOS et WINDOWS :

- OK : la réponse est disponible dans le buffer de la requête.
- PENDING : pas encore de réponse, à réessayer plus tard.
- sinon, un code d'erreur négatif :

  ENOK : Il y a une erreur générale remontée par le driver. Consultez la variable globale **usExtinfo** pour en connaître la cause. (cf:Tables des erreurs).

En retour OS/2 :

OK : La réponse est disponible dans le buffer de la requête
MPW_RETRY : Pas encore de réponse, à réessayer plus tard
sinon un code d'erreur négatif :

PBINITSESSION	Contexte de communication on initialisé
MPW_CLIENTNOK	Canal de communication non alloué
MPW_REQPENDING	Requête en cours sur le canal de communication
MPW_SEQ	Problème enchaînement sur serveur
NORESAVAIBLE	Pas ou plus de ressource disponible
MPW_REQABORTED	Requête abandonnée
PBDRV	Problème driver Prise Terminal
RESPREF	Code réponse refusé
MPW_ERR_SYSTEM	Erreur système

6.7-2 UNITEResponseMult (OS/2 uniquement)

Description :

Fonction demande de compte-rendu sur une liste de requêtes.

Syntaxe :

```
UNITE_RC rc  =  UNITEResponseMult   (PHREQ     phRequête,
                                    short      ulMod
                                    PUSHORT    pusReponse);
```

En entrée :

phReq : pointeur de la liste des identificateurs de requêtes (hReq) dont on demande des réponses.
Cette liste doit se terminer par HREQ_NULL (-1).

ulMode : mode de fonctionnement.

- WAIT : réponse demandée avec attente infinie.
- ABANDON : abandon des requêtes de la liste.
- autre valeur : temps d'attente enveloppe de la réponse (en ms.).

En sortie:

pusReponse : lorsque la réponse d'une requête est arrivée, pusReponse pointe l'identificateur hReq de la fonction correspondante.
Si plusieurs réponses sont arrivées, pusReponse pointera l'identificateur de la première réponse reçue par le serveur.
Remarque : l'ordre de réception des réponses par le serveur est indépendant de l'ordre dans lequel les fonctions UNITE_XX ont été appelées.

En retour :

- OK : une des réponses de la liste des requêtes est arrivée (son indice dans la liste est pointé par pusReponse). Pour obtenir les réponses des autres requêtes, réitérer la demande en enlevant la requête terminée du tableau.
- WAIT : pas encore de réponse, à réessayer plus tard.
- sinon, un code d'erreur négatif :

PBINITSESSION	Contexte de communication on initialisé
MPW_CLIENTNOK	Canal de communication non alloué
MPW_REQPENDING	Requête en cours sur le canal de communication
MPW_SEQ	Problème enchaînement sur serveur
NORESAVAIBLE	Pas ou plus de ressource disponible
MPW_REQABORTED	Requête abandonnée
PBDRV	Problème driver Prise Terminal
RESPREF	Code réponse refusé
MPW_ERR_SYSTEM	Erreur système

7.1 Lexique

Canal de communication : ressource permettant d'émettre les requêtes (liées à des fonctions) à destination d'un équipement distant.

Client UNI-TE : équipement capable d'émettre des requêtes UNI-TE.

Couches : Cf "**Modèle OSI** : modèle de référence structurant toute communication en 7 *couches* distinctes, communiquant chacune avec leurs voisines. Ces différentes couches ont pour nom et fonction :

- 7	APPLICATION	fenêtre entre des processus d'application dans le but d'échanger des informations significatives
- 6	PRESENTATION	représentation d'informations circulant entre des processus d'application
- 5	CESSION	organisation et synchronisation du dialogue entre deux processus d'application et gestion de l'organisation de leurs données
- 4	TRANSPORT	transfert d'informations entre deux systèmes de manière transparente et fiable
- 3	RESEAU	routage des informations entre deux équipements
- 2	LIAISON	transfert d'informations entre deux systèmes adjacents avec détection d'erreurs
- 1	PHYSIQUE	transmission d'éléments binaires entre deux systèmes via un médium de communication

UNI-TE : système de messagerie client-serveur de TELEMECANIQUE.

Serveur UNI-TE : équipement répondant aux demandes de services (requêtes) émises par un équipement client.

Serveur minimum : équipement ne supportant en serveur que les requêtes UNI-TE obligatoires : *Mirror, Identification, Protocol_Version et Status.*

Transaction : ensemble d'une requête et de sa réponse.

«Txt mod»

7.2 Liste des codes d'erreur DOS et WINDOWS

Toutes les fonctions de la librairie retournent un code. Cette valeur permet de contrôler l'intégrité de l'échange de bout en bout.
Elle est disponible dans le mot status.

7.2-1 Codes communs à toutes les fonctions

OK	(0)	
ENOK	(-1)	: Erreur générale (précision dans usExtinfo)
PENDING	(-2)	: Attente
EREFUSEDFLAG	(-3)	: Flag invalide
EBORNES	(-4)	: Valeur hors bornes
EDRVNOTOPEN	(-5)	: Le driver n'a pas été ouvert
EMEMFULL	(-6)	: Il n'y a plus de mémoire libre
ESRCEADR	(-7)	: Adresse source invalide
EDESTADR	(-8)	: Adresse destination invalide
ECONNOTOPEN	(-9)	: Canal de communication non ouvert
EBUILD	(-10)	: Erreur à la création du datagramme
EBADMODE	(-11)	: Erreur sur le paramètre mode
ERESPREF	(-12)	: Réponse UNI-TE négative
EMIRROIR	(-13)	: La requête Mirror s'est mal déroulée
ERQTIMEOUT	(-14)	: Erreur timeout sur la requête
EBADADD	(-15)	: Erreur mauvaise adresse
ENOMORESOCK	(-16)	: Erreur max canal de communication par driver atteint
ENOTOPENSOCK	(-17)	: Erreur canal de communication non alloué
EDRVAOPEN	(-18)	: Erreur driver déja ouvert
EBADBUFFER	(-19)	: Erreur buffer pSendRequest non initialisé
EBADPARAM	(-20)	: Erreur paramètre invalide
ECLEPROT	(-21)	: Erreur clé non présente
EUNSOOPEN	(-22)	: Canal des données non sollicitées déjà ouvert

7.2-2 Valeurs relatives aux fonctions de transfert de fichier

ENOTSXV4	(-30)	: L'automate distant n'est pas >= V4
ETYPTSXDIFF	(-31)	: Le processeur du fichier et celui du distant ne sont pas identiques
ECOMPATINOK	(-32)	: L'indice du distant et celui du fichier ne sont pas compatibles
EROMCART	(-33)	: La cartouche dans le TSX est une ROM
ENOVALIDCART	(-34)	: La cartouche dans le TSX n'est pas valide
EAPIOUTMEM	(-35)	: La taille mémoire du TSX est trop petite
ETSXETATNOK	(-36)	: Le TSX ne permet pas le téléchargement
EDATAORDER	(-37)	: Erreur cohérence des données
ESEQ	(-38)	: Erreur de séquencement
ETSXWRITE	(-39)	: Ecriture impossible
EVERIFAPPNOK	(-40)	: Vérification de l'API charge impossible
EFICH	(-41)	: Erreur à l'ouverture du fichier
ELECFICH	(-42)	: Erreur à la lecture du fichier
ENOFICHAPP	(-43)	: Le fichier n'est pas un programme TSX
ENAKENTRESERV	(-44)	: Erreur à l'entretien de réservation
EAPIPROT	(-45)	: Erreur API protegée
ENOAPI	(-46)	: Erreur pas d'API dans l'automate
EECRFICH	(-47)	: Erreur écriture du fichier
ETYPEINVAL	(-48)	: Erreur type de fichier invalide
EPROGEXIST	(-49)	: Erreur Programme déjà présent
EMANIPMEMCN	(-50)	: Erreur Manipulation mémoire CN
ESATURMEM	(-51)	: Erreur Saturation mémoire
EPBFILE	(-52	: Erreur Problème fichier
ECNNORAZ	(-53)	: Erreur Etat CN non en RAZ
ESIZE	(-54)	: Erreur Size incohérente
ESTCNINCOMP	(-55)	: Erreur Etat CN incompatible
ETIMEOUTUCCN	(-56)	: Erreur Time out Coprocessseur
EFICHFERME	(-57)	: Erreur fichier déja fermé
ECLE	(-58)	: Erreur de clé ou numéro d'affaire
eabortoper	(-59)	: Séquence interrompue par l'opérateur
EVERIF	(-60	: Erreur en vérification
EENDFILE	(-61)	: Erreur fin de fichier
ECRFICH	(-62)	: Erreur à l'écriture du fichier

7.2-3 Codes d'erreurs liés à la variable usExtinfo

ENOERROR	(0)	
ENETUNREACH	(1)	: Driver ou Réseau non accessible
EHDLOUTRANGE	(2)	: Pas de connexion pour le handle
ESOCKNOTOPEN	(3)	: Canal de communication non alloué
ESOCKOUTOFR	(4)	: Numero de canal de communication hors limi-
tes		
ESOCKNOMORE	(5)	: Pas de canal de communication utilisable
ESOCKOPENED	(6)	: Canal de communication déjà réservé
ESOCKNOEMPTY	(7)	: Messages en attente sur ce canal de comm.
EPRNOSUPPORT	(8)	: Protocole non supporté par le driver
EINVLEN	(9)	: Buffer size invalide
ENOTCONN	(10)	: Connexion impossible
EFULL	(11)	: Les buffers du driver sont tous utilisés
EBADFLAG	(12)	: Mode asynchrone non supporté
ETIMEOUT	(13)	: Timeout dépassé
ENOASYNCH	(14)	: IOCB non alloué en mode asynchrone
ESEMOUTOFRANGE	(15)	: Adresse d'un sémaphore invalide
EINVALIDMODE	(16)	: Mode flag invalide
EATTEMPTINGLINKUP	(17)	: Driver tente de rétablir la communication
EBADPARAMETER	(18)	: Mauvais paramètre dans IOCB
ELOCALNACK	(19)	: La carte n'a pas renvoyé d'acquit.
EREMOTENACK	(20)	: L'équipement n'a pas renvoyé d'acquit
EBADVERSION	(21)	: Réseau et driver incompatibles
EUNKNOWNERROR	(22)	: Erreur inconnue

7.3 Liste des codes d'erreur OS/2

Toutes les fonctions de la librairie retournent un code. Cette valeur permet de contrôler l'intégrité de l'échange de bout en bout.
Elle est disponible dans le mot status.

7.3-1 Codes communs à toutes les fonctions

MPW_OK	(0)	Tout est OK
MPW_NO_OK	(-1)	Problème liaison
MPW_RETRY	(-2)	La réponse n'est pas encore arrivée
TIME_OUT	(-3)	Dépassement du temps enveloppé pour un évchange.
NACK	(-4)	Le message a été reçu, mais faute de ressources, il n'a pas été traité par le destinataire.
BORNE	(-5)	Index hors bornes.
PBINIT	(-6)	Driver non initialisé.
PBRECEPTRAME	(-7)	Problème sur réception de message.
NORESAVAIBLE	(-8)	Plus de ressource disponible.
PBINITSESSION	(-9)	Pas de canal initialisé sur cet équipement.
PBSHUTDOWN	(-10)	Problème destruction queue.
RESPREF	(-11)	Code réponse refusée reçu.
INITNOK	(-12)	Code réponse Init refusée.
MIRRORNOK	(-13)	Code réponse Mirror refusée.
NOKEY	(-14)	Clé de protection absente.

7.3-2 Valeurs relatives aux fonctions de transfert de fichier

ERRFICH	(-20)	Ouverture fichier impossible
NOFICHAPP	(-21)	Le fichier n'est pas un fichier Application
ERRECRFICH	(-22)	Erreur écriture fichier
ERRLECFICH	(-23)	Erreur lecture fichier
NAKRESERV	(-24)	Réservation TSX impossible
NOTSXV4	(-25)	Le TSX distant n'est pas de type V4
TYPTSXDIFF	(-26)	Les TSX fichier et distant sont différents
TSXETATNOK	(-27)	L'état du TSX est incompatible avec le chargement
COMPATINOK	(-28)	Problème de compatibilité TSX
APIPROT	(-29)	Application protégée
NOAPI	(-30)	Application inexistante
ROMCART	(-31)	La cartouche est une ROM
NOVALIDCART	(-32)	Cartouche non valide
APIOUTMEM	(-33)	Application trop importante pour TSX distant
NAKSTOP	(-34)	Impossible de stopper le TSX distant
ERRSEQ	(-35)	Erreur de séquencement
ERRTSXWRITE	(-36)	Ecriture du TSX impossible
ERRDATAORDER	(-37)	Données incohérentes
NAKENTRESERV	(-38)	Erreur entretien réservation
ENDSEQNAK	(-39)	Fin de séquence de déchargement refusée
VERIFAPPNOK	(-40)	Erreur après vérification Appli non valide

7.4 Requêtes UNI-TE

7.4-1 Lecture d'objets

Cette requête permet la lecture d'objets simples (mots ou chaîne de mots...) par la fonction UNITEReadObject.

Format de la requête

Code requête H/D	Code catégorie	Segment	Type d'objet	Adresse de l'objet		Nombre d'objets à lire	
36/54	0 → 7						

Segment : spécifie le mode d'adressage des objets à lire ainsi que l'espace où ils se trouvent (en hexadécimal).
Les segments accessibles par les automates TSX série 7 sont (en hexadécimal) :

- 10 : segment des objets communs,
- 64 : segment espaces bits internes,
- 68 : segment espace mots internes,
- 69 : segment espace mots constants,
- 6C : segment des tâches utilisateurs Ctrl,
- 80 : segment des objets système TSX 7
- 81 : segment des blocs fonctions,
- 82 : segment des modules d'entrées / sorties.

Type d'objet : spécifie le type d'objet à lire :

- 0 : bloc texte ou module en bac,
- 1 : bloc Ctrl,
- 5 : bits internes avec forçage,
- 7 : entier signé 16 bits,
- 8 : entier signé 32 bits,
- 64 : période d'une tâche.

Adresse de l'objet :
- adresse physique ou logique dans le segment.
- numéro d'ordre de l'objet dans le segment :
  - 0 : date et heure courantes dans le segment commun,
  - 1 : date et heure sauvegardées dans le segment commun, (dernier passage en Stop)
  - 2 : date et heure courantes (en hexadécimal) dans le segment commun,

Lecture d'objets (suite)

Format du compte rendu

Compte rendu positif

Code réponse H/D	Type d'objet	Données				
66/102						

Type d'objet : retourne le type d'objet choisi lors de l'envoi de la question.

Compte rendu négatif

Code réponse H/D
FD/253

Causes de rejet :
- Requête inconnue,
- Droits d'accès insuffisants,
- Segment ou objet inconnu,
- Adresse hors bornes,
- Nombre d'objets trop important pour le buffer de réception.

Exemples de requêtes

Lecture mots ou doubles mots

Segment : 68 (segment des mots internes),
Type d'objet : 7 → Wi ou 8 → DWi,
Adresse de l'objet : indice du premier Wi ou DWi à lire,
Réponse : tableau de n objets.

Lecture mots constants ou doubles mots constants

Segment : 69 (segment des mots constants),
Type d'objet : 7 → CWi ou 8 → CDWi,
Adresse de l'objet : indice du premier CWi ou CDWi à lire.
Réponse : tableau de n objets.

Lecture d'objets (suite)

Lecture date et heure

Segment : 10 (segment des objets communs),
Type d'objet : 0 par défaut,
Adresse de l'objet : 0 → date et heure courantes,
1 → date et heure sauvegardées,
Quantité : 0 par défaut,
Réponse : adresse de l'objet = 0 (date et heure courantes) :
AAAAMMJJHHMMSS.DN
adresse de l'objet = 1 (date et heure sauvegardées) :
AAAAMMJJHHMMSS.DP
AAAA = année,
MM = mois,
JJ = jour,
HH = heure,
MM = minute,
SS = seconde,
D = dixième de seconde,
N = jour de la semaine,
P = code de la coupure secteur.

Lecture période d'une tâche

Segment : 6C (segment des tâches utilisateurs Ctrl),
Type d'objet : 64 (période d'une tâche),
Adresse de l'objet : 1 → tâche interruption,
2 → tâche rapide,
3 → tâche maître,
4 → tâche auxiliaire 0,
5 → tâche auxiliaire 1,
6 → tâche auxiliaire 2,
7 → tâche auxiliaire 3,
Quantité : 0 par défaut,
Réponse : période de la tâche codée sur un octet (1 à 255) en respectant les bases de temps de chaque tâche (FAST = 1ms, MAST = 1 ms et AUXi = 10 ms). Pour la tâche IT, la réponse correspond au nombre de cycles d'EXEC déclenchés.

Lecture bits internes

Segment : 64 (segment des bits internes),
Type d'objet : 5 (bits internes avec forçage),
Adresse de l'objet : numéro logique du premier bit interne,
Quantité : nombre de bits à lire modulo 8,
Réponse : tableau de n bits contenant l'état des bits suivi d'un autre tableau de n bits indiquant si le bit correspondant est forcé ou non.

Lecture d'objets (suite)

Lecture date et heure (en hexadécimal)

Segment : 10 (segment des objets communs),
Type d'objet : 0 par défaut,
Adresse de l'objet : 2 → date et heure courantes en hexadécimal,
Quantité : 0 par défaut,
Réponse : Tableau de huit mots indiquant :
les millisecondes,
les secondes,
les minutes,
l'heure,
le jour,
le mois,
l'année,
le numéro du jour dans la semaine.

Lecture des paramètres d'un bloc fonction texte

Segment : 81 (segment des blocs fonctions),
Type d'objet : 0 (bloc texte),
Adresse de l'objet : numéro logique du premier bloc texte,
Quantité : nombre de blocs texte consécutif à lire,
Réponse : tableau de bits et mots indiquant pour chaque bloc texte :

TXTi,D	:	bit (1 = done),
TXTi,E	:	bit (1 = erreur),
Indirect	:	bit (1 = bloc texte indirect),
Distant	:	bit (1 = bloc texte distant),
Non défini	:	4 bits non significatifs,
Type	:	0 = TXT, 1 = CPL, 2 = TER, 3 = SYS, 5 = TLG,
TXTi, A	:	mot,
TXTi, M	:	mot,
TXTi, T	:	mot,
TXTi, C	:	mot,
TXTi, R	:	mot,
TXTi, S	:	mot,
TXTi, L	:	mot.

Les blocs textes mis à jour dans les tâches IT ou FAST risquent d'être lus avec des valeurs apparemment incohérentes. Ceci est du au fait que cette requête est traitée dans la tâche Maître qui est moins prioritaire que les tâches IT ou FAST.

Lecture d'objets (suite)

Lecture d'un bloc CTRL

Segment : 81 (segment des blocs fonctions),
Type d'objet : 1 (bloc Ctrl),
Adresse de l'objet : numéro logique du premier bloc Ctrl,
Quantité : nombre de blocs Ctrl consécutifs,
Réponse : Tableau d'octets structuré comme suit :

Etat tâche
- non configurée : 0
- en STOP : 1
- en RUN : 2
- point d'arrêt : 3
- défaut logiciel : 4

tâche active : Bit 0 = active,
Bit 1 à 7 non significatifs,

période : 0 à 255. Pour la tâche IT, ce champ correspond au nombre d'activation de cette tâche depuis l'initialisation de l'application.

Lecture d'un module d'entrées / sorties en bac

Segment : 82 (segment des modules d'entrées / sorties),
Type d'objet : 0 (module en bac),
Adresse de l'objet : adresse du module définie comme suit :
- bits 8 à 11 : numéro de station,
- bits 3 à 6 : numéro de bac,
- bits 0 à 2 : numéro de module,

les autres bits sont non significatifs,

Quantité : 1,
Réponse : Tableau d'octets structuré comme suit :
- octet de défaut : se reporter à la requête lecture de l'image mémoire d'un module d'entrées / sorties,
- octet de configuration : se reporter à la requête lecture de l'image mémoire d'un module d'entrées / sorties,
- octet indiquant le code d'extension configuré,
- octet indiquant l'état physique (bit 0 = erreur d'acquittement, bit 1 = erreur de parité, les autres bits sont non significatifs),
- octet indiquant le code d'extension du module physique.

7.4-2 Ecriture d'objets

Cette requête permet l'écriture d'objets simples (mots ou chaîne de mot...) par la fonction UNITEWriteObject.

Format de la requête

Code réquête H/D	Code catégorie	Segment	Type d'objet	Adresse de l'objet		Nombre d'objets à écrire		Données
37/55	0 → 7							

Segment : spécifie le mode et le champ d'adressage (en hexadécimal) :

- 10 : segment des objets communs,
- 64 : segment des espaces bits internes,
- 68 : segment des espaces mots internes,
- 69 : segment des espaces mots constants,
- 6C : segment des tâches utilisateurs Ctrl.

Type d'objet : spécifie le type d'objet à écrire :

- 5 : bits internes,
- 7 : entier signé 16 bits
- 8 : entier signé 32 bits,
- 64 : période d'une tâche.

Adresse de l'objet :

- adresse physique ou logique dans le segment.
- numéro d'ordre de l'objet dans le segment :
  - 0 : date et heure courantes dans le segment commun,
  - 1 : configuration de la prise terminal dans le segment système.

Format du compte rendu

Compte rendu positif

Code réponse H/D
FE/254

Ecriture d'objets (suite)

Compte rendu négatif

Code réponse
FD/253

Causes de rejet :
- Requête inconnue,
- Droits d'accès insuffisants,
- Objet inconnu,
- Adresse du dernier objet hors bornes.

Exemples de requêtes

Ecriture mots ou doubles mots

Segment : 68 (segment des mots internes),
Type d'objet : 7 → Wi ou 8 → DWi,
Adresse de l'objet : indice du premier Wi ou DWi à écrire,
Quantité : nombre,
Données : tableau de n objets.

Ecriture mots constants ou doubles mots constants

Segment : 69 (segment des mots constants),
Type d'objet : 7 → CWi ou 8 → CDWi,
Adresse de l'objet : indice du premier CWi ou CDWi à écrire.
Quantité : nombre,
Données : tableau de n objets.

Ecriture date et heure

Segment : 10 (segment des objets communs),
Type d'objet : 0 par défaut,
Adresse de l'objet : 0 → date et heure courantes,
Quantité : 0 par défaut,
Données : 17 caractères ASCII décrivant la date et l'heure :
AAAAMMJJHHMMSS.DN,
AAAA = année,
MM = mois,
JJ = jour,
HH = heure,
MM = minute,
SS = seconde,
D = dixième de seconde,
N = jour de la semaine,

Ecriture période d'une tâche

Segment : 6C (segment des tâches utilisateurs Ctrl),
Type d'objet : 64 (période d'une tâche),
Adresse de l'objet : 2 → tâche rapide,
3 → tâche maître,
4 → tâche auxiliaire 0,
5 → tâche auxiliaire 1,
6 → tâche auxiliaire 2,
7 → tâche auxiliaire 3,
Quantité : 0 par défaut.
Données : nouvelle période de la tâche en respectant les bases de temps de chaque tâche (FAST = 1ms, MAST = 1 ms et AUXi = 10 ms).

Lecture bits internes

Segment : 64 (segment des bits internes),
Type d'objet : 5 (bits internes),
Adresse de l'objet : numéro logique du premier bit interne,
Quantité : nombre de bits à écrire modulo 8,
Données : tableau d'octets contenant l'état des bits, chaque octet représente la valeur de huit bits (le forçage des bits ne peut pas être écrit).

7.4-3 Initialisation application

Cette requête permet la commutation vers l'état Stop d'un automate en défaut logiciel par la fonction générique UNITERequest.

Format de la requête

Code requête H/D	Code catégorie
EF/239	0 → 7

Attention : Selon le type de produit, la réservation préalable peut être nécessaire.

Format du compte rendu

Compte rendu positif

Code requête H/D
FE/254

Compte rendu négatif

Code requête H/D
FD/253

Causes de rejet :
- Requête inconnue,
- Droits d'accès insuffisants,
- Non réservation.