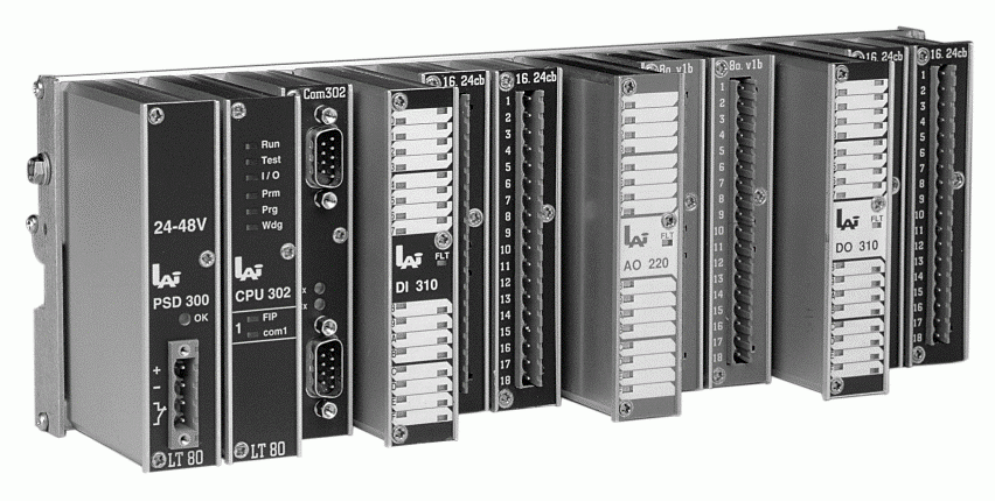
leroy AUTOMATION

PLATE-FORME D'AUTOMATISME LT80 - LT160 ATELIER ISaGRAF

LT ISaGRAF

MANUEL UTILISATEUR



P DOC ISA 001 F – V2.0

FOURNITURES

Le Kit **LT ISaGRAF** comprend :

- Le présent manuel utilisateur du LT ISaGRAF.
- 1 câble RS232 permettant la connexion du PC au LT.
- 1 disquette : "Librairies LT ISaGRAF".

Le manuel de mise en oeuvre du LT (80 ou 160) est fourni avec le LT.

Cette documentation présente les fonctionnalités de :

- **logiciel embarqué (noyau) :**
  - ⇒ **CPU non ethernet : version 2.01**
  - ⇒ **CPU ethernet : version 1.86**
- **librairies LT ISaGRAF : version 2.0**

SUPPORT TECHNIQUE :
Tél : 33.(0).5.62.24.05.46
Fax : 33.(0).5.62.24.05.55
e-mail : support@leroy-autom.com

Prérequis : La mise en œuvre du LT ISaGRAF nécessite les compétences suivantes : connaissance de l'atelier ISAGRAF et des langages d'automatisme IEC 61131.

Leroy Automatique Industrielle
Siège : Boulevard du Libre échange
31650 Saint Orens
Tél : 33.(0).5.62.24.05.50
Fax : 33.(0) 5.62.24.05.55
Site internet : http://www.leroy-automation.com

SOMMAIRE

1 Présentation générale

Le **LT (LT80 ou LT160)** de Leroy Automation est programmable avec l'atelier ISaGRAF de Altersys. Le LT est alors appelé **LT ISaGRAF**.

Cette documentation présente la mise en oeuvre de l'atelier ISaGRAF sur la cible LT. L'installation de l'atelier ISaGRAF ainsi que son utilisation (création d'un projet, programmation...) sont détaillées dans le "Guide utilisateur" d'ISaGRAF fourni avec l'atelier.

Le schéma suivant offre une vue d'ensemble de l'architecture ISaGRAF sur le LT :

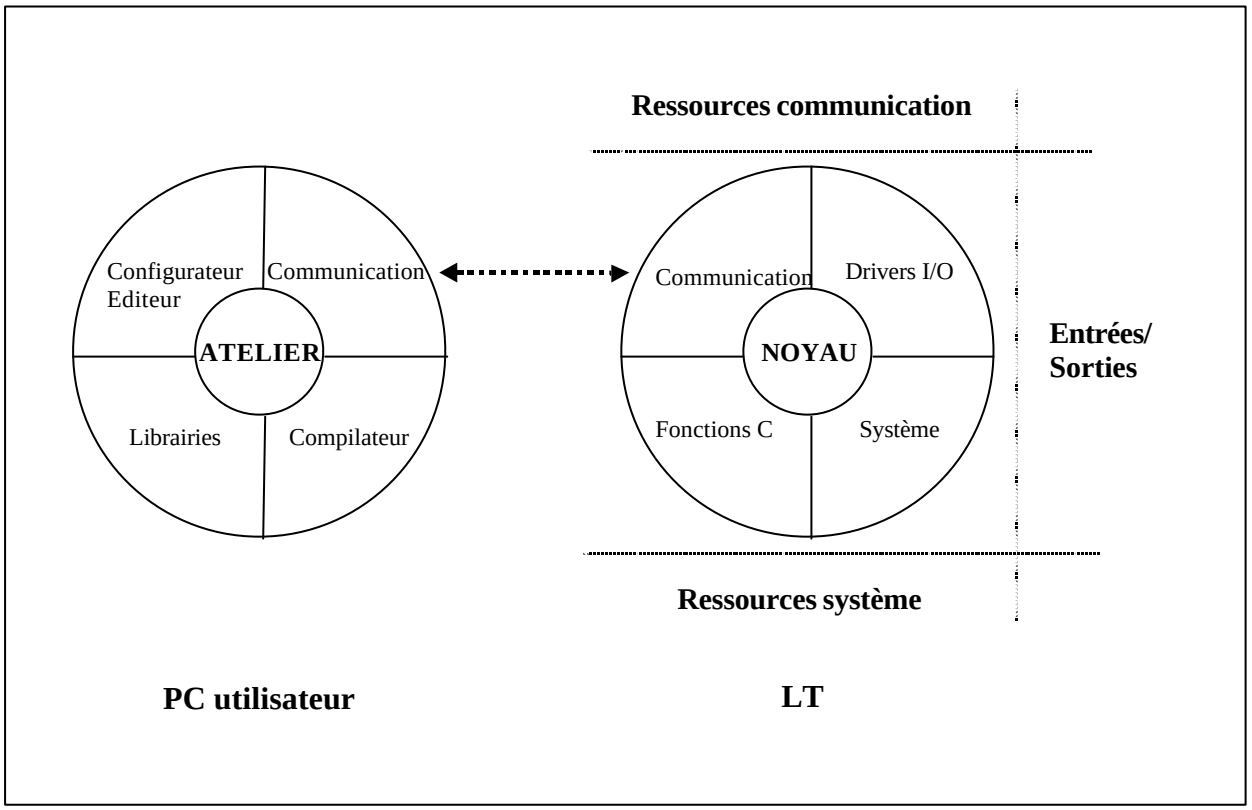


Figure 1 : architecture ISaGRAF sur le LT

La **cible LT** supporte le **noyau ISaGRAF** et permet l'accès aux ressources suivantes :

- système : horloge, mémoire...
- drivers d'entrées/sorties
- drivers de communication.

L'atelier ISaGRAF permet de créer et modifier de nouveaux projets à destination de la cible. Avec l'utilitaire "**Librairies**", l'utilisateur dispose de fonctions permettant de gérer le LT :

- insertion de cartes d'entrées/sorties,
- insertion de drivers de communication (Modbus, Jbus, Ethernet),
- fonctions de pilotage des entrées/sorties (leds),
- fonctions de diagnostic du LT (lecture du status des cartes...).

1.1 Ressources matérielles du LT

Le LT est une base matérielle de travail. Sur cette base, le noyau ISaGRAF va s'exécuter et utiliser les ressources matérielles disponibles. C'est pourquoi, une présentation, même succincte du matériel, peut aider à la compréhension des modes de fonctionnement du LT et des fonctions associées.

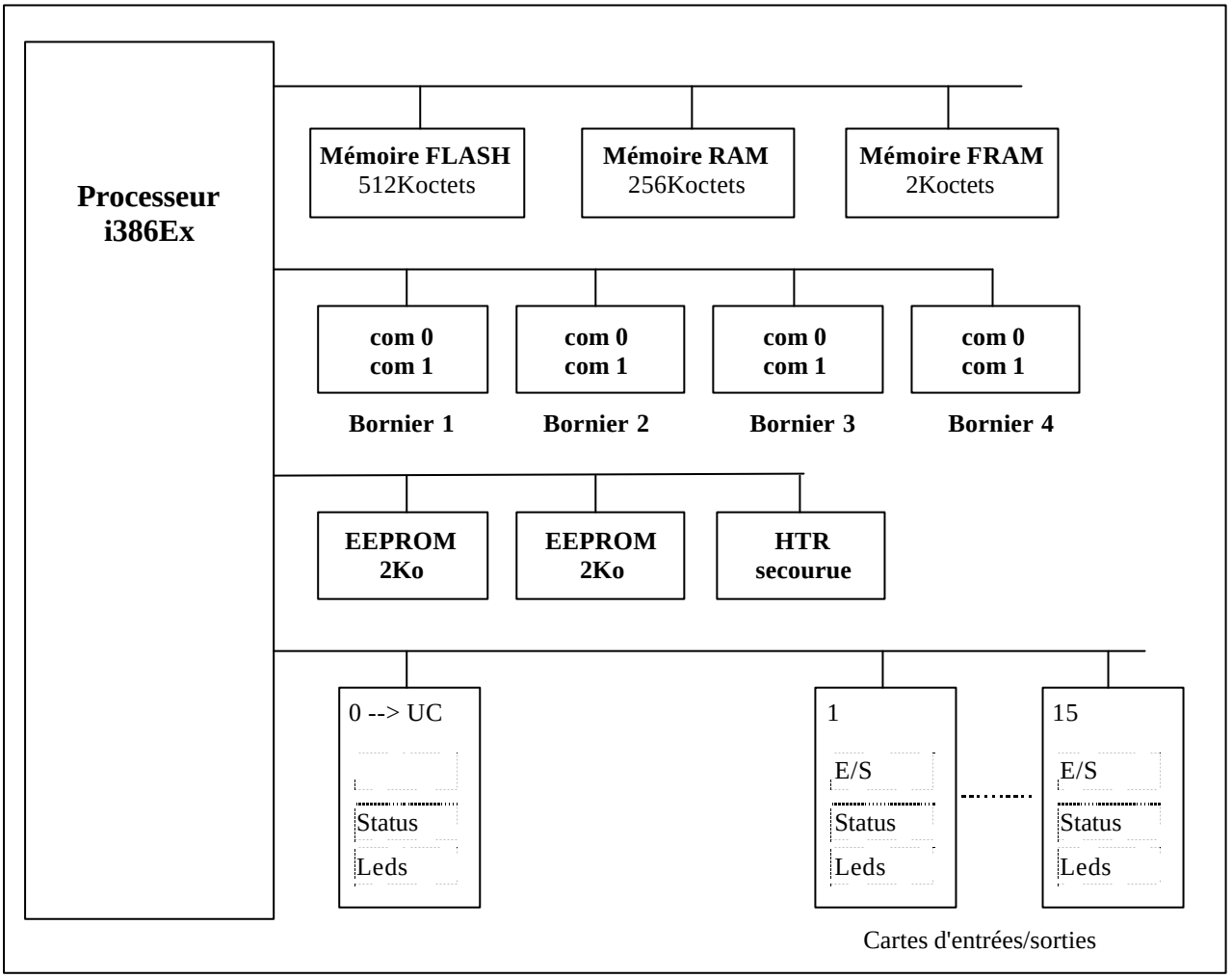


Figure 2 : ressources matérielles LT160

Nota : Le **LT80** comporte les limitations suivantes :

- 128 Koctets de mémoire RAM,
- 1 seul bornier de communication (com 1 et com 0),
- 3 emplacements de cartes d'entrées/sorties,
- 1 seule EEPROM.

1.2 Cycle des traitements effectués

Suivant le schéma bien connu d'un automate, le noyau ISaGRAF exécute le cycle de traitements suivant:

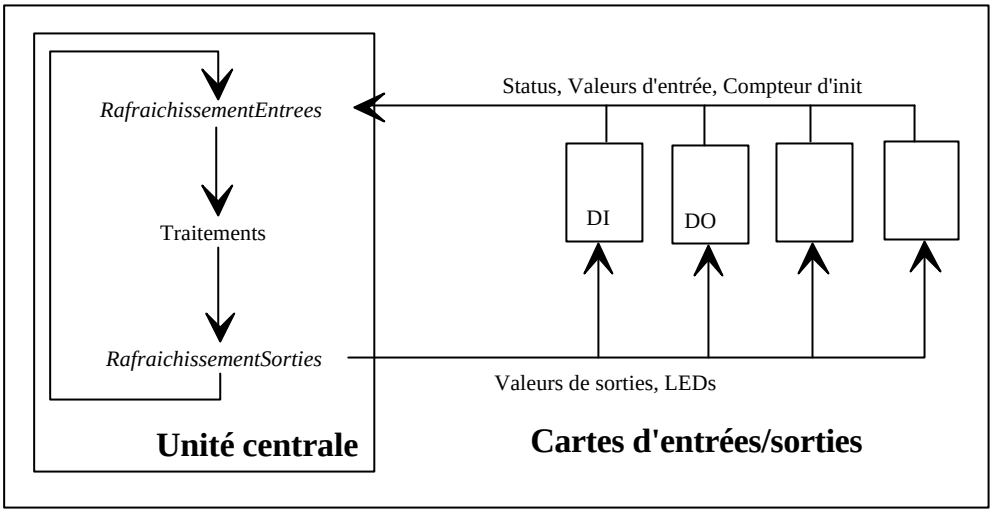


Figure 3 : cycle de traitements du LT ISaGRAF

2 Première mise en oeuvre

2.1 Installation de l'atelier ISaGRAF

A partir du "Guide utilisateur" d'ISaGRAF, suivez les instructions du chapitre "Mise en route". La configuration matérielle et logicielle requise par l'atelier ISaGRAF est suffisante pour fonctionner avec la cible LT.

Lancer l'atelier ISaGRAF.

2.2 Intégration des librairies et projets spécifiques au LT dans l'atelier ISaGRAF

Leroy Automation a développé des **librairies (ou bibliothèques) spécifiques au LT,** ainsi que **des exemples de projets**. Elles permettent d'exploiter les ressources spécifiques au LT : modules d'entrées/sorties, protocoles de communication distante...

Ces librairies et projets sont fournis sur la disquette « Librairies LT ISaGRAF ».

2.2.1 Décompression des fichiers sources

Exécuter le fichier « LibrairiesLtIsagraf.exe » présent sur la disquette : choisissez un nouveau répertoire sur votre disque dur, par exemple « C:\Isawin\Lai » et lancer la décompression.

Le répertoire que vous avez choisi contient alors des fichiers correspondant à des objets ISaGRAF :

Nature	Type de l'objet	Extension de fichier
Librairies	Configurations d'E/S	*.ria,
	Cartes d'E/S	*.bia
	E/S complexes	*.xia
	Fonctions	*.iia
	Blocs fonctionnels	*.aia
	Fonctions C	*.uia
	Blocs fonctionnels C	*.fia
Projets		*.pia

2.2.2 Intégration des librairies spécifiques au LT dans l'atelier ISaGRAF

Lancer le gestionnaire de librairies.

Dans le menu "**Fichier**"/"**Autres librairies**", sélectionner un type de librairie : « **Cartes d'E/S** » par exemple.

Dans le menu "**Outils**"/"**Archiver**" : sélectionner chaque élément contenus dans la liste "**Archive**" et cliquez sur le bouton "**Restituer**". Ne pas oublier de sélectionner dans l'unité de sauvegarde le répertoire « C:\Isawin\Lai » ou celui que vous aviez choisi lors de l'étape de décompression.

Répéter cette opération pour chaque type d'objet en les sélectionnant tour à tour, à partir du menu "**Fichier**"/"**Autres librairies**", ou de la liste déroulante, sous la barre de menu, et en répétant l'opération de restitution précédente.

Une fois tous les objets intégrés dans l'atelier, toutes les librairies spécifiques au LT pourront être utilisées comme des librairies ISaGRAF standard.

L'utilitaire "Librairies" fourni avec l'atelier ISaGRAF et utilisé pour générer ces fonctions permet de visualiser la "**Fiche technique**" spécifique à chaque fonction. Elle contient tous les renseignements nécessaires à l'utilisation des fonctions ou cartes d'entrées/sorties (paramètres, code de retour, restrictions, exemples...).

2.2.3 Intégration des projets spécifiques au LT dans l'atelier ISaGRAF

Lancer le gestionnaire de projets.

La même opération de restitution que pour les Librairies est à réaliser pour les exemples de projets depuis le gestionnaire de projets : dans le menu "**Outils**"/"**Archiver**"/"**Projets**", sélectionner chaque élément contenus dans la liste "**Archive**" et cliquez sur le bouton "**Restituer**".

Liste des projets disponibles :

- Lt232sig : mise en œuvre des signaux RS232 des borniers Com311
- Lte2prom : mise en œuvre de la lecture et de l'écriture des E2prom
- LThtr : mise en œuvre de l'horloge sauvegardée
- Ltjbusm1 : mise en œuvre du protocole modbus asynchrone maître
- Ltjbusm2 : mise en œuvre du protocole modbus asynchrone maître avec utilisation de grafcet fils
- Ltjbuss : mise en œuvre du protocole modbus asynchrone esclave
- Ltledio : mise en œuvre du pilotage des leds des cartes d'entrées et sorties
- Ltio : mise en œuvre du câblage des cartes d'entrées et sorties, lecture de status, compteurs d'initialisation
- LtNulcar : mise en œuvre d'une communication simple avec traitement du caractère nul
- LtNulpro : mise en œuvre d'une communication simple : émission et réception d'octets.
- Ltpid : mise en œuvre d'un bloc fonctionnel pid
- Ltsaveva : mise en œuvre des variables non volatiles
- LtTCPe : mise en œuvre du protocole modbus/TCP esclave
- LtTCPm : mise en œuvre du protocole modbus/TCP maître
- Lttemail : mise en œuvre du protocole SMTP : envoi d'email
- LTxmulti : exemple de mise en œuvre simultanée d'un ensemble de protocoles de communication : modbus/TCP (maître et esclave), modbus asynchrone (maître et esclave).
- LTxomod : exemple de la fonction passerelle réseau Ethernet / réseau asynchrone : protocoles modbus/TCP esclave et modbus asynchrone maître

2.3 Une première application

On peut réaliser un programme minimal "qui ne fait rien" de la manière suivante :

- créer un nouveau projet : "Fichier"/"Nouveau Projet" dans la fenêtre "Gestionnaire de projets",
- choisir la "Configuration d'E/S" : LT ou positionner "cpu3xx" sur le slot 0 du câblage des entées/sorties,
- générer l'application,
- télécharger dans le LT.

Ce programme minimal ne gère aucune ressource d'entrées/sorties du LT et n'effectue aucun traitement.

Important : la **liaison console** (ou liaison atelier) se trouve par défaut sur le **com1 du bornier 1**. Elle peut être placée sur tout autre port de communication.

Dans les prochains paragraphes, nous détaillerons les traitements spécifiques que l'on peut programmer sur le LT ISaGRAF.

2.4 Configuration matérielle

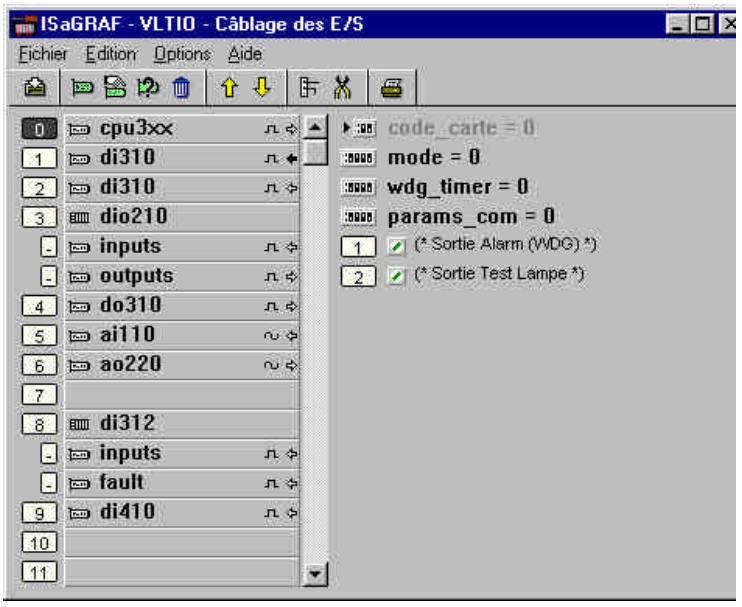
La configuration matérielle du LT est réalisée à partir du menu **"Outils"/"Câbler les E/S".** Cette fenêtre propose 256 emplacements (slots) de cartes d'entrées/sorties. Seuls les 16 premiers peuvent être utilisés pour programmer un LT. Pour un LT80 seuls les 4 premiers slots sont utilisables : 1 CPU + 3 cartes d'E/S.

Le premier slot est réservé à l'unité centrale ou CPU : CPU3xx. ou **CPUETH.**

L'alimentation ou PSD (PSD300 ou PSD331) ne se configure pas graphiquement. Elle est déclarée automatiquement par le noyau.

Les ports de communications associés à chaque CPU sont déclarés dans le programme ISaGRAF à l'aide de fonctions spécifiques.

Le rack du LT est représenté de manière verticale et les cartes d'entrées/sorties numérotées de 0 à 15. Voici un exemple de configuration et sa correspondance sur le LT :



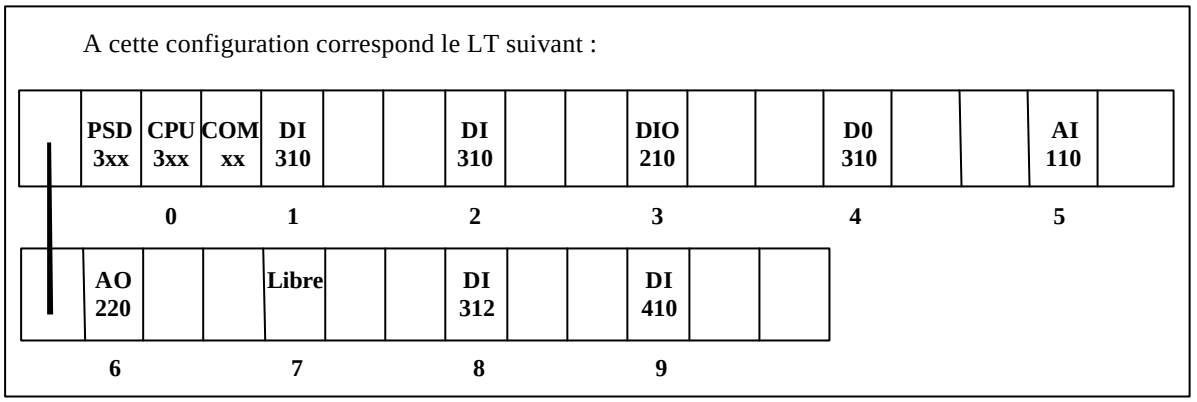


Figure 4 : principe de câblage des entrées/sorties

Sous chaque carte du LT est noté son rang logique : CPU = 0, 1°E/S = 1, ... , 15° E/S = 15. Noter que ce rang correspond à la numérotation de l'éditeur de câblage ISaGRAF. La carte DIO210 est un équipement composé de plusieurs cartes. Il ne prend qu'un slot sur la liste des cartes déclarées. De même pour la carte DI312.

Nota : les blocs d'extension ne se configurent pas. Dans l'atelier de câblage, un LT est vu comme un seul rack.

2.5 Connexion de l'atelier ISaGRAF au LT

Lorsque vous mettez un LT ISaGRAF sous tension, il exécute l'algorithme suivant :

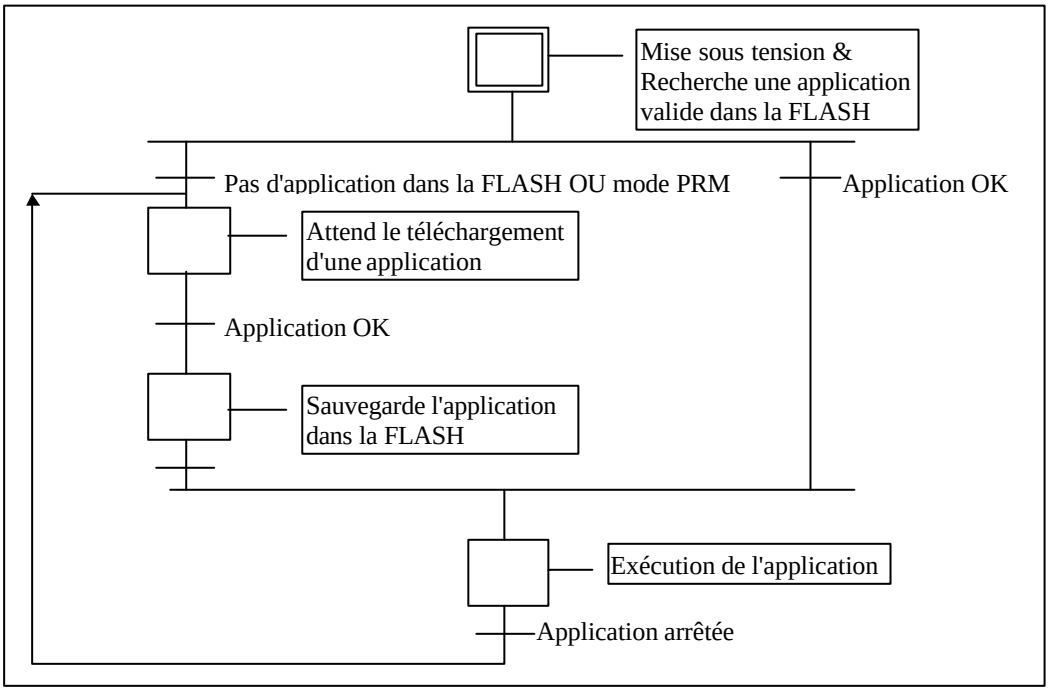


Figure 5 : principe d'exécution d'un LT ISaGRAF

La Led "RUN" clignote lentement (période 1 sec) lorsque l'application s'exécute sur le LT.

Pour se connecter au LT, il est nécessaire de réaliser les étapes suivantes :

- connecter le câble RS232 fourni entre un COM du PC et le **com1 du bornier 1** du LT,
- créer et générer une nouvelle application sur l'atelier. La génération s'effectue à partir du menu **"Codage"/"Générer l'application"**. Dans la liste "Options de compilation" sélectionner seulement "**TIC code for INTEL**" *.

- dans le menu tester, configurer les paramètres de communication : **esclave 1, 19200 bps, sans parité, 1 bit de stop, 8 bits de données et sans contrôle de flux**. Régler par exemple le time-out à 10 secondes et le nombre d'essais à 3. Enfin choisir le port de communication utilisé sur le PC.
- choisir le menu **"Test"/"Tester".**

*** TIC** = Target Independent Code : code dédié aux noyaux ISaGRAF.

Suivant le grafcet vu précédemment deux cas se présentent :

- une application n'a pu être récupérée dans la FLASH : le bandeau du débogueur affiche **"Pas d'application**",
- une application a été récupérée dans la FLASH : le bandeau du débogueur affiche **" 'nom de l'appli' active"**. Les informations sur le temps de cycle et l'état de cette application apparaissent alors dans le bandeau principal du débogueur.

Consulter le chapitre A.16 "**Mise au point**" du "Guide utilisateur" ISaGRAF afin d'utiliser correctement le débogueur.

Si la connexion ne peut pas s'établir entre le LT et l'atelier, on peut forcer l'atelier à ne pas récupérer d'application en FLASH : c'est le **mode paramétrage du LT**.

2.6 Téléchargement d'une application

Le téléchargement d'une application de l'atelier vers le LT s'effectue à partir du menu du débogueur et de la commande **"Fichier"/"Transférer"**.

Important : la récupération d'une application dans le LT ISaGRAF n'est pas possible.

Pour pouvoir effectuer le transfert, il est nécessaire d'avoir le **message "Pas d'application"** dans le bandeau du débogueur :

- soit en étant en mode paramétrage,
- soit en arrêtant l'application active depuis le débogueur.

Choisir transférer pour débuter le téléchargement de l'application. Le bandeau du débogueur indique le pourcentage de l'application transférée. A la fin du transfert, l'application est automatiquement sauvegardée en FLASH par le LT. L'application est alors exécutée en mode temps réel.

Si une erreur se produit lors de l'écriture en Flash elle sera remontée vers l'atelier sous la forme d'un numéro allant de 100 à 255 (cf. paragraphe Erreurs 6.2).

2.7 Mise au point d'une application

Une application peut être mise au point de deux manières :

- sur le PC avec le simulateur accessible par le menu "Test"/"Simuler",
- sur le LT avec le débogueur accessible par le menu "Test"/"Tester".

L'utilisation de ces deux modes de mise au point est détaillée dans le "Guide utilisateur" d'ISaGRAF.

La taille du code TIC généré par l'atelier ISaGRAF correspond à la taille du fichier **appli.x8m**. Ce fichier se trouve dans le répertoire \isawin\apl\"nom application"\.

La taille d'une application est limitée à 64Ko : fichier appli.x8m < 64Ko.

3 La carte CPU

3.1 Le paramétrage de la carte CPU

La carte CPU se décline en 2 versions : **cpu3xx et cpueth**. Elle doit obligatoirement se trouver sur le **premier slot** de l'éditeur de câblage des E/S.

La **cpu3xx** : carte CPU du LT 80 ou LT 160 **sans** port Ethernet.
La **cpueth** : carte CPU du LT80 ou LT 160 **avec** un port Ethernet.

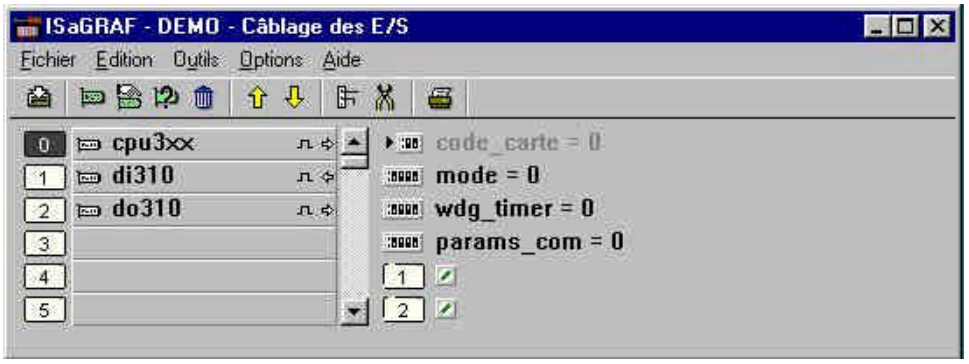
Paramètres communs à ces deux versions :

- **code_carte** : il est figé : égal à 0 pour la cpu3xx, égal à 256 pour la cpueth.
- **mode** : permet de configurer des fonctionnements différents du noyau (type Word, 0 par défaut),
- **wdg_timer** : permet de borner le temps de cycle du LT (en millisecondes). (type Word, 0 par défaut),
- **params_com** : paramètres de communication de la liaison console. (type Long Hexa, 0 par défaut).

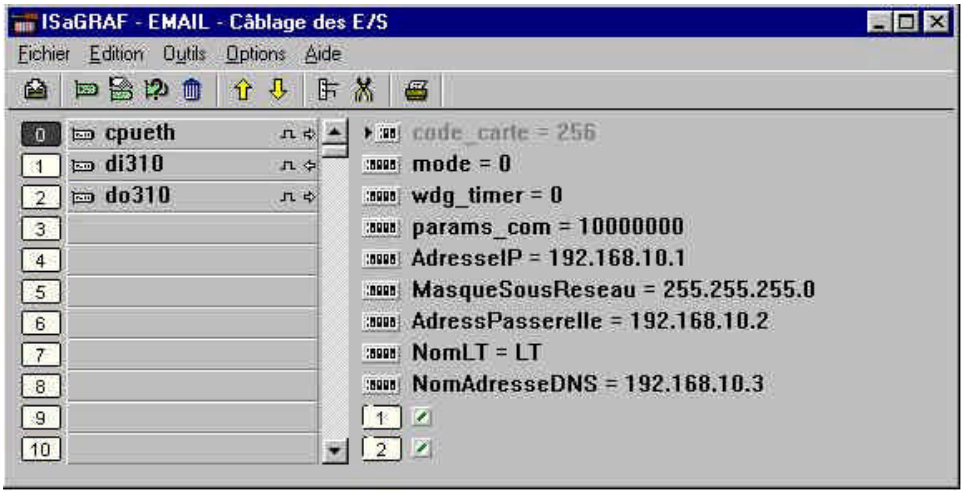
Cette carte dispose également de **deux sorties booléennes** représentant :

- la **sortie Alarme** de l'alimentation. Le pilotage de la sortie Alarme inhibe la gestion et le rafraîchissement des E/S,
- la **sortie Test Lampe** permettant de vérifier si les leds des cartes entrées/sorties fonctionnent. Le pilotage de cette sortie allume toutes les leds des cartes d'entrées/sorties.

Exemple de paramétrage de la carte cpu3xx :



Exemple de paramétrage de la carte cpueth :



Paramètres supplémentaires liés à la version cpueth :

- **Adresse IP** : identifie le réseau et l'équipement (l'automate LT) sur un réseau TCP/IP.

  Par défaut l'adresse IP vaut 255.255.255.255. Dans ce cas le LT va ignorer les autres paramètres et utiliser un serveur d'adresse BOOTP : celui-ci renverra au LT une adresse IP disponible sur le réseau.

  Format : xxx.xxx.xxx.xxx avec xxx [0..255]

- **Masque Sous Réseau** : masque d'adresse permettant de mettre en évidence la division de l'adresse IP en adresse de réseau et sous-réseau et adresse de l'équipement sur ce sous-réseau. Ce masque de 32 bits comporte des 1 pour les parties de l'adresse de réseau et sous-réseau et des 0 pour les parties de l'adresse de l'équipement.

  Format : xxx.xxx.xxx.xxx avec xxx [0..255]

- **Adresse Passerelle** : adresse IP de la passerelle se trouvant sur le réseau. Si le LT veut communiquer en dehors du réseau auquel il appartient, il doit s'adresser à cette passerelle. Par défaut, cette adresse vaut 127.0.0.1 et identifie le LT lui-même (pas de passerelle).

  Format : xxx.xxx.xxx.xxx avec xxx [0..255]

- **Nom LT** : nom symbolique du LT (à utiliser pour la connexion au serveur SMTP) :

  Format : 10 caractères alphanumériques maximum.

- **Adresse DNS** : Adresse IP du serveur DNS (*Domain Name Server*). Ce serveur retourne une adresse IP à partir d'un nom symbolique désignant un équipement ou un serveur sur un réseau TCP/IP. Il est utile pour les connexions au serveur SMTP et dans la gestion du protocole modbus / TCP maître.

  Format : xxx.xxx.xxx.xxx avec xxx [0..255]

Paramètres relatifs au protocole SNMP (librairie Cpuetsnm) :

- **Num_Agent** : Numéro d'agent SNMP dans la sous-branche LAI(4273) ; par défaut à 0 : service SNMP inactif

- **AdrIP_Manager** : Adresse IP du manager SNMP : seules les requêtes de ce manager sont traitées ; par défaut à 255.255.255.255 : les requêtes de tout manager SNMP sont traitées

- **Emplacement** : renseigne le champs « location » dans la MIB II du LT

Community : « public » par défaut ; il peut être personnalisé : le LT répondra uniquement aux requêtes envoyées par un manager de sa communauté.

3.1.1 Paramètre « mode » : modes de fonctionnement disponibles

- bit 0 : **mode chien de garde (WDG)**
  - à 0 : le chien de garde est géré par le noyau (défaut)
  - à 1 : le chien de garde est géré par l'application ISaGRAF (code TIC)

    la sortie 1 sur la carte cpu doit être cablée : dans le programme applicatif, si elle est forcée à :
    - false : pas de Wdg
    - true : les cartes d'entrées/sorties ne sont plus rafraîchies.
- bit 1 : **mode Secure**
  - à 0 : aucun contrôle de présence (défaut)

- à 1 : toute carte présente sur le rack doit se trouver dans l'atelier de câblage sinon une erreur IO est signalée et le WDG n'est pas désactivé.

- bit 2 : **mode FreeIO**
  - à 0 : une carte d'E/S se trouvant dans l'atelier de câblage et absente sur le rack génère une erreur IO (défaut)
  - à 1 : une carte d'E/S se trouvant dans l'atelier de câblage peut être absente sur le rack sans générer d'erreur IO.

3.1.2 Paramètre « params_com » : paramètres de la liaison console

Par défaut, pour les 2 versions cpu3xx et cpueth, la liaison console est sur le port série du bornier 1 com1 ; Par défaut, params_com=0 et les paramètres de communication sont :

- bornier 1
- com 1
- esclave 1
- vitesse 19200 bauds
- parité sans
- bit stop 1
- donnés 8 bits

la liaison console peut être paramétrée à n'importe quel autre emplacement des borniers liés à la CPU, aussi bien sur un port série que sur le port ethernet.

Pour **modifier les paramètres de communication** il faut changer la valeur de params_com. Chaque paramètre est codé sur 1 quartet de params_com, le quartet 1 étant le quartet de poids faible :

Bornier	Com	N° esclave		Données	BitsStop	Parité	Vitesse
Quartet 8	Quartet 7	Quartet 6	Quartet 5	Quartet 4	Quartet 3	Quartet 2	Quartet 1

Le **paramétrage par défaut** (params_com = 0) correspond aussi à **params_com = 1101810A**

3.1.2.1 Liaison console sur un port série RS232

Le codage des paramètres qui correspondent aux paramètres de l'atelier ISaGRAF est le suivant :

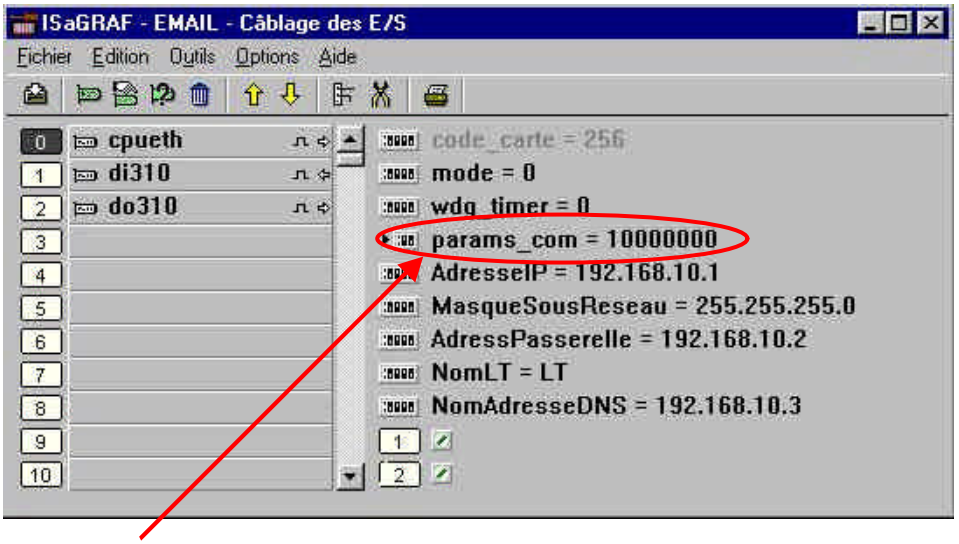
Numéro du Quartet	Valeur du Quartet	Valeur du paramètre associé
Quartet 1 (vitesse)	5	600 Bauds
	6	1200 Bauds
	7	2400 Bauds
	8	4800 Bauds
	9	9600 Bauds
	A	19200 Bauds
Quartet 2 (parité)	0	Pas de parité
	1	Parité paire
	2	Parité impaire
Quartet 3 (bit de stop)	1	1 bit de stop
	2	2 bit de stop
Quartet 4 (données)	7	7 bits
	8	8 bits
Quartet 5 et 6 (n° d'esclave)	[0..FF]	[0..255]
Quartet 7 (Com)	0	Com0
	1	Com1

Quartet 8 (Bornier)	1	Bornier 1
	2	Bornier 2
	3	Bornier 3
	4	Bornier 4

3.1.2.2 Liaison console sur le port ethernet

Ce paramétrage n'est possible que sur la version cpueth.

Pour réaliser cette opération il suffit de modifier le paramètre params_com dans la fenêtre de câblage comme indiqué ci-après :

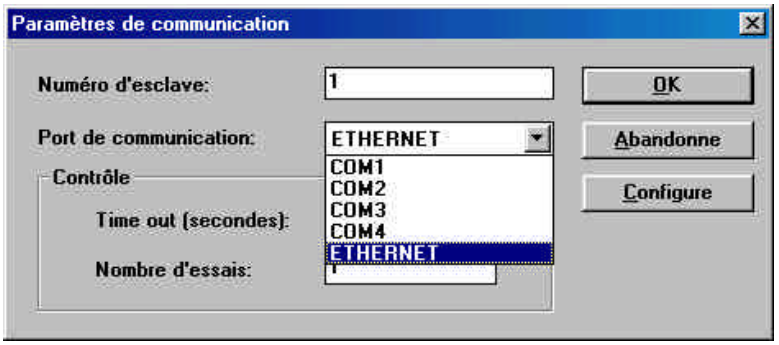


params_com=10000000 désigne le com0 du bornier1 (port Ethernet) comme liaison console.

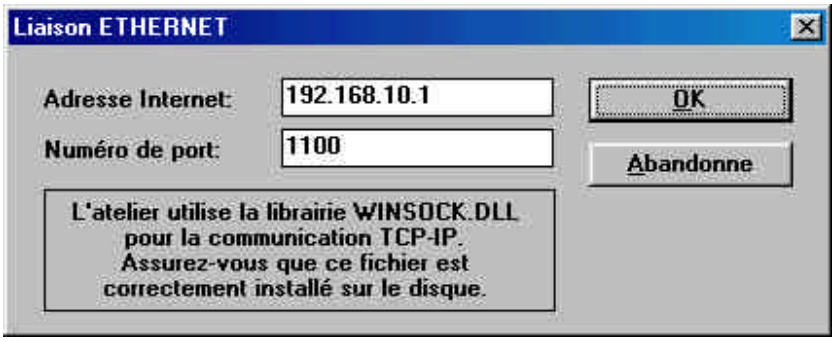
Dans le cas du port Ethernet, le numéro d'esclave est remplacé par l'adresse IP du LT et le format de communication est imposé par la norme IEEE 802.3 (10Base-T).

Après avoir modifié le params_com qui agit sur le LT, généré et téléchargé l'application dans l'automate, il faut sélectionner le port de communication correspondant dans l'atelier ISaGRAF. Pour cela cliquer sur «Configurer la liaison » du menu «Test » dans la fenêtre « Programmes »

La fenêtre ci-dessous apparaît. Sélectionner « Ethernet » dans le menu déroulant « Port de communication ». Le numéro d'esclave de ce port doit être « 1 ». Cliquer maintenant sur le bouton « Configure ».



Remplir le Champ « Adresse Internet » avec l'adresse IP de votre LT.



La led Com0 bornier 1 présente sur le module CPU du LT Ethernet signale la surveillance sur la connexion avec l'atelier ISaGRAF. Allumée fixe lorsque l'atelier ISaGRAF est connecté sur le LT Ethernet, cette led clignote lorsque la liaison est rompue après un timeout de 2 secondes.

3.2 Sauvegarde des données :

3.2.1 Accès à l'EEPROM du LT

Le LT dispose de deux EEPROM dans le cas d'un LT160 et une seule dans la cas d'un LT80. Dans l'unité centrale, l'accès à ces composants s'effectue à l'aide d'une communication série via le protocole I2C. Chaque composant est adressable par son numéro qui est respectivement 4 et 5. La taille d'une EEPROM est de 1kmots (1024 mots). Elles sont accessibles en lecture et en écriture d'entiers (mots de 16 bits).

Exemple d'écriture puis de lecture d'un mot dans l'EEPROM n° 4 : **voir l'exemple de projet « lte2prom »**

- *Status := E2p_W(4, 2, ValeurAEcrire);*
- *ValeurLue = E2p_R(4, 2*);

Status est une variable booléenne, *ValeurAEcrire* et *ValeurLue* sont des variables entières (seuls les 16 bits de poids faible sont significatifs).

Nota : la variable entière écrite sera bornée à [-8000h..7FFFh].

Fonction	**E2p_R**
ACTION	Lit 1 mot dans une EEPROM du LT
SYNTAXE	*entier Donnee E2p_R(entier NumE2p, entier AdrLecture);*
PARAMETRES	NumE2p : [4, 5] sur le LT80 le n° est 4 AdrLecture : [0..1023]
VALEUR RENVOYEE	Donnée: valeur entière lue [0..FFFFh] si erreur : 10000h
EXEMPLE	*Donnee := E2p_R(5, 3);*

Fonction	E2p_W
ACTION	Ecrit 1 mot dans une EEPROM du LT
SYNTAXE	*booleen Status E2p_W(entier NumE2p, entier AdrEcriture, entier Valeur);*
PARAMETRES	NumE2p : [4,5] sur le LT80 le n° est 4 AdrEcriture : [0..1023] Valeur : [0..FFFFh]
VALEUR RENVOYEE	Status : [TRUE, FALSE]
EXEMPLE	*Status := E2p_W(5, 3, 16#0F0F);*

3.2.2 Variables non volatiles

Le LT dispose d'une **mémoire secourue** d'une taille de **2048 octets**. Pour secourir une variable en cas de coupure d'alimentation du LT, il suffit de **cocher la case "non volatile"** lors de la déclaration de la variable. Avec un LT, il n'est pas nécessaire de configurer les paramètres d'exécution d'une application ISaGRAF comme spécifié dans le Guide Utilisateur ISaGRAF.

Sur les 2048 octets de la mémoire secourue, **1980 sont réservés à la sauvegarde des données non volatiles**. L'espace occupé par type de variable est le suivant :

- 1 octet par variable booléenne,
- 4 octets par variable entière + 4 octets pour l'ensemble des variables entières
- 5 octets par variable temporisation
- 1 octet par caractère d'une variable message + 3 octets par variable message

La seule contrainte est : si 1 variable non volatile est cochée, il est nécessaire d'en cocher une de chaque type. 4 types de variables peuvent être non volatiles : booléen, entier, temporisation et message.

Le principe de sauvegarde et récupération d'une variable secourue est le suivant :

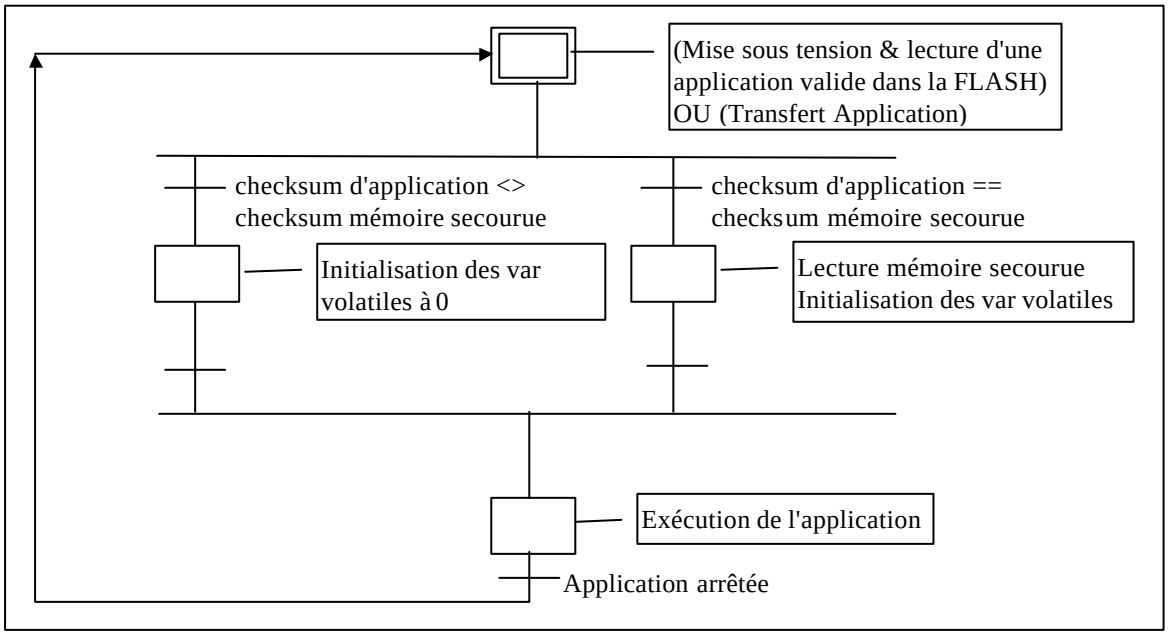


Figure 6 : principe de traitement des variables non volatiles

3.2.3 L'horloge secourue

Le LT ISaGRAF dispose d'une **horloge logicielle secourue**. Cette horloge **fournit la date, l'heure et le jour de la semaine**. Ces données peuvent être lues ou écrites par l'utilisation de fonctions C dans l'atelier ISaGRAF .

Fonction	**DayTim_O**
ACTION	Initialise l'accès à l'horloge sur le LT
SYNTAXE	*booleen Status **DayTim_O**();*
PARAMETRES	Aucun
VALEUR RENVOYEE	Status : TRUE=initialisation correcte FALSE=erreur d'initialisation
DESCRIPTION	Une seule initialisation est nécessaire pour un projet.
EXEMPLE	Status := DayTim_O();

Fonction	**DayTim_W**
ACTION	mise à l'heure (date et heure) de l'horloge sur le LT
SYNTAXE	*booleen Status **DayTim_W**(entier TypeInfo, message Chaine);*
PARAMETRES	TypeInfo : 0 : date 1 : heure 2 : jour semaine Chaine : message suivant le type d'info modifiée :
VALEUR RENVOYEE	Status : TRUE=initialisation correcte FALSE=erreur d'initialisation
FORMATS	date : AAAA/MM/JJ heure : HH:MM:SS.CC jour : Dimanche = 0 ; Lundi = 1 ; Mardi = 2 ; Mercredi = 3 ; Jeudi = 4 ; vendredi = 5 ; samedi = 6
EXEMPLE	**Ecriture ou mise à jour** d'une date, heure et jour semaine. Par exemple le vendredi 31 décembre 2001 à 23 heures 58 minutes 10 secondes et 02 centièmes : Booleen1 := DayTim_W(0, '2001/12/31'); Booleen2 := DayTim_W(1, '23:58:10.02'); Booleen3 := DayTim_W(2, '5');

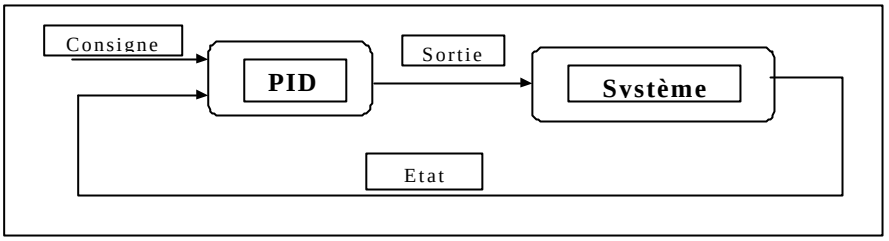
Fonction	**Day_Time**
ACTION	Donne la date ou l'heure ou le jour sous forme de chaîne de message
SYNTAXE	*message Status Day_Time(entier TypeInfo);*
PARAMETRES	**cf guide utilisateur**
VALEUR RENVOYEE	**cf guide utilisateur**
DESCRIPTION	**FONCTION standard d'ISaGRAF : cf guide utilisateur** Le LT ISaGRAF fournit en plus les centièmes de secondes. L'heure a donc le format suivant : HH:MM:SS.CC
EXEMPLE	**cf guide utilisateur**

Exemple : voir l'exemple de projet « lthtr ». Egalement, un bloc fonctionnel « horloge » permet de régler l'horloge à partir d'un maître modbus(voir la fiche d'aide de ce bloc).

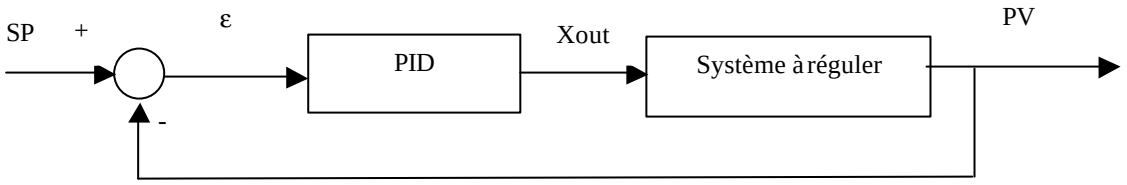
3.3 Fonction de régulation : PID

3.3.1 Principe et mise en œuvre sur le LT ISaGRAF :

Un PID est un régulateur de système fonctionnant sur le principe du retour d'état. La sortie du système est régulée en utilisant la différence entre la sortie réelle du système et l'état qu'elle devrait avoir. Ce principe est résumé par le schéma bloc ci-après.



Le modèle du PID implanté sur le LT ISaGRAF est :



Avec SP = Consigne, PV = Valeur de retour

$Xout(t)=Kp(e(t)+\frac{1}{Ti}\int_0^t e(t)d(t)+Td\frac{d\ e\ (t)}{dt})$ en continue soit

$Xout(k)=Kp(e(t)+\frac{Ts}{Ti}I(k)+\frac{Td}{Ts}(e(k)-e(k-1)))$ en discret avec $I(k)=I(k-1)+e(k)Ts$.

Caractéristiques du bloc fonctionnel C PID_AL :

Bloc fonctionnel	**PID_AL**	
ACTION	Initialise, paramètre et rafraîchit un bloc PID	
SYNTAXE	*Instance (*booleen Auto, reel Pv , reel Sp, reel X0, reel Kp, reel Ti, reel Td, temporisation Ts, reel Min, reel Max*);* *Commande :=Instance.Xout;*	
PARAMETRES	Auto	mode de fonctionnement du bloc : automatique : Auto=True manuel : Auto=False
	Pv	valeur de retour du process
	Sp	Consigne
	X0	valeur de réglage sortie pid pour le mode manuel
	Kp	gain proportionnel général à toutes les actions
	Ti	constante d'intégration en s
	Td	constante de dérivation. en s
	Ts	période d'échantillonnage en ms
	Min, Max	valeurs limites basses et hautes pour la sortie
VALEUR DE RETOUR	Xout	valeur de sortie
EXEMPLE	pid1 est une instance de pid_al (dictionnaire, onglet « instances BF ») pid1(Auto, Pv , Sp, X0, Kp, Ti, Td, Ts, Min, Max); Sortie :=pid1.xout ;	

Le traitement se décompose en trois actions: proportionnelle, intégrale, dérivée. Chaque action est réglable séparément ainsi on peut composer une régulation à partir des trois actions.

La mise en oeuvre d'un PID est réalisée par l'utilisation d'un bloc fonctionnel C Pid_AL() :

- déclarer une instance de PID : par exemple "pid1" dans le dictionnaire,
- suivant la période d'échantillonnage, appeler l'instance avec ses paramètres,
- exemple : pid1(Auto1, Pv1 ,Sp1, X01, Kp1, Ti1, Td1, Ts1, Min1, Max1);
- sortie1 est une variable numérique,
- la valeur de sortie renvoyée est dans : sortie1 := pid1.Xout;

voir le projet de démonstration : «ltpid.pia »

Il est possible de composer un régulateur P, PI, PD, PID. Pour cela, il suffit de désactiver l'action qui n'est pas utilisée. Une action (proportionnelle, intégrale ou dérivée) est désactivée lorsque les paramètres dont elle dépend sont : Kp=1 ou Ti=0 ou Td=0.

3.3.2 Méthode de réglage

Le réglage du régulateur PID passe par le choix des paramètres Kp, Ti, Td. Pour déterminer les paramètres Kp, Ti, Td des méthodes d'analyse expérimentale du procédé sont possibles.

Par exemple, les spécifications typiques pour les appareils conduisant des processus chimiques ou thermiques sont les suivants:

- Ti de 3 à 1000 secondes,
- Td de 3 à 150 secondes.

Une méthode de réglage en ligne : la méthode par essai-erreur.

Le réglage en ligne peut se faire de façon empirique en utilisant une procédure qu'on peut résumer ainsi:

- mettre la régulation en place,
- enlever l'action intégrale et dérivée,
- mettre le gain Kp à une faible valeur,
- faire une petite variation de la consigne et observer la réponse du système. Comme le gain est très petit, la réponse sera très amortie,
- doubler le gain et refaire l'étape précédente. Continuer ainsi de suite jusqu'à ce que la réponse devienne oscillante. Appelons cette valeur Kpu (ultimate Kp),
- mettre Kp à (Kpu / 2),
- faire la même opération en réduisant Ti par un facteur 2, jusqu'à obtenir une réponse oscillante pour une petite variation de la consigne,
- mettre Ti au double de cette valeur,
- procéder de même pour la constante dérivée: augmenter Td jusqu'à obtenir une réponse oscillante, puis mettre Td à 1/3 de cette valeur.

4 Gestion des communications

Le LT 160 dispose de 1 à 8 liaisons séries ; le LT80 de 1 à 2 liaisons séries.

Le **com1 du bornier 1** (com1 des borniers Com301 et Com303) **est par défaut la liaison console**. Cette liaison console peut cependant être placée sur un autre port. La liaison console permet le dialogue avec l'atelier ISaGRAF (téléchargement, débogueur...) ou l'accès aux variables du dictionnaire par un maître Modbus/Jbus (cf. paragraphe suivant). **Le com1 du bornier 1 supporte seulement une liaison RS232.** Une voie de communication est obligatoirement dédiée à la liaison console.

Le **com0 du bornier Com303 est une liaison Ethernet** et ne peut pas être configuré autrement. C'est la seule liaison Ethernet possible.

Les autres liaisons RS232 ou RS485 (com1 des Com301 et Com303, com0 et com1 des borniers Com311 et Com312) peuvent supporter soit un protocole **JBus maître ou esclave** , soit un protocole spécifique **d'émission/réception d'octets**, soit la **liaison console** si nécessaire.

Sur le LT160, 4 emplacements pour borniers de communication sont disponibles. Si l'on dispose de la liaison Ethernet, on peut configurer 6 liaisons séries + 1 liaison console. Si l'on n'a pas la liaison Ethernet, on peut configurer 7 liaisons séries + 1 liaison console.

Les différents types de borniers sont les suivants :

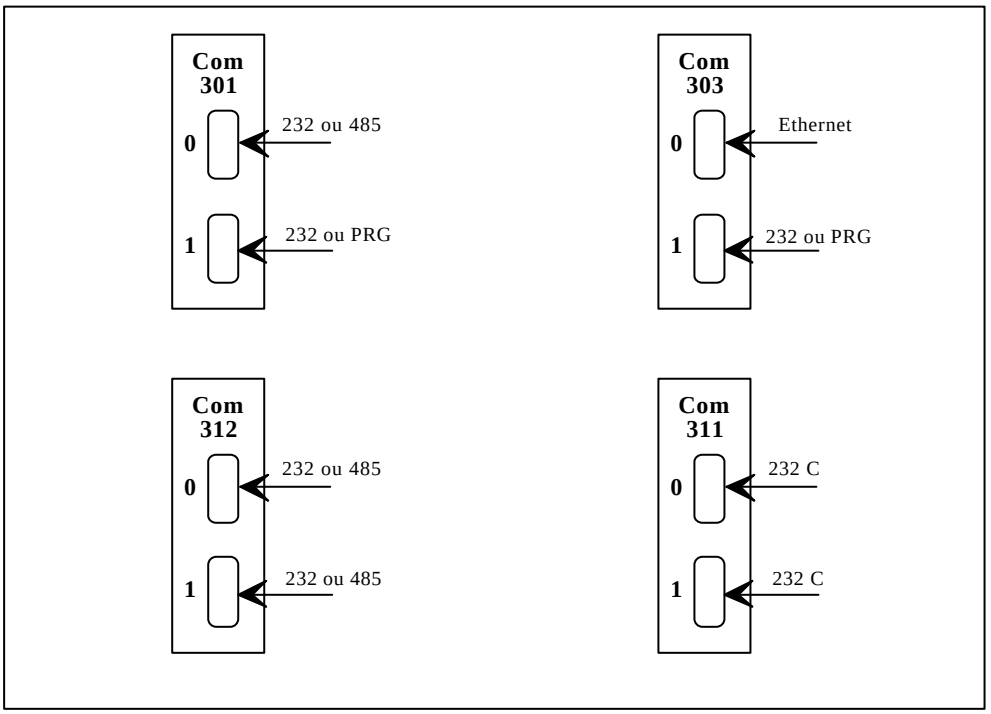


Figure 7 : borniers de communication du LT

Sur le bornier 1, on trouve obligatoirement un Com301 ou un Com302. Un seul Com311 peut être utilisé et se trouve obligatoirement sur le bornier 2.

Sur le LT, la communication est gérée à travers **2 couches logicielles** :

La **couche basse** est particulière à chaque liaison série. Elle stocke les octets reçus, détecte une fin de trame par dépassement du temps de silence, émet la réponse éventuelle du LT. Elle est réalisée sous interruption (spécifique à la liaison série) et transparente vis à vis du programme utilisateur.

La **couche haute** est indépendante des liaisons série. Elle analyse la trame reçue, effectue éventuellement le travail demandé par le maître et prépare la réponse à émettre. Cette couche est traitée par les fonctions utilisateur spécifiques à chaque protocole.

Protocoles disponibles :

Protocoles sur réseau RS232 et RS485 :

- Jbus Esclave,
- Jbus Maître,
- Protocole simple d'émission/réception.

Attention : le choix de la liaison série (RS232 ou RS485) dépend uniquement du câblage réalisé.

Protocoles sur réseau ethernet :

- ModBus/TCP :
  - esclave : interrogation par 1 ou plusieurs maîtres ModBus/TCP
  - maître : interrogation d'un ou plusieurs esclaves Modbus/TCP
- SMTP : envoi de courrier électronique

4.1 Principe de communication sur les ports de communication

Les ports série RS232/485 ou RS232C supportent soit le **protocole Modbus/Jbus,** soit un protocole spécifique bâti sur **l'émission/réception d'octets**, soit la **liaison console** si nécessaire.

Le principe d'utilisation du protocole Modbus/Jbus sur un port du LT est le suivant :

initialiser (ou déclarer) une voie de communication Modbus/Jbus Esclave ou Maître revient à utiliser une fonction C spécifique depuis l'atelier. Cette fonction va définir les paramètres de communication sur la voie et y associer une table d'échange de n mots.

Les trames venant d'un maître ou esclave rafraîchiront cette table d'échange. Les données de chaque table peuvent être utilisées depuis l'atelier (variables du dictionnaire) par des fonctions spécifiques :

- ***Word_R()*** : lit sous forme non signée un mot d'une table vers une variable entière du dictionnaire,
- ***Word_W()*** : écrit une variable entière du dictionnaire vers un mot d'une table,
- ***Bit_R()*** : lit un bit d'une table vers une variable booléenne du dictionnaire,
- ***Bit_W()*** : écrit une variable booléenne du dictionnaire vers un bit d'une table,
- ***DWord_R()*** : lit deux mots d'une table vers une variable entière du dictionnaire,
- ***DWord_W()*** : écrit une variable entière du dictionnaire vers deux mots d'une table,
- ***WordS_R*** : lit sous forme signée un mot d'une table vers une variable entière du dictionnaire.

Les paramètres sont détaillés ci-après.

Exemple pour un protocole Jbus Esclave :

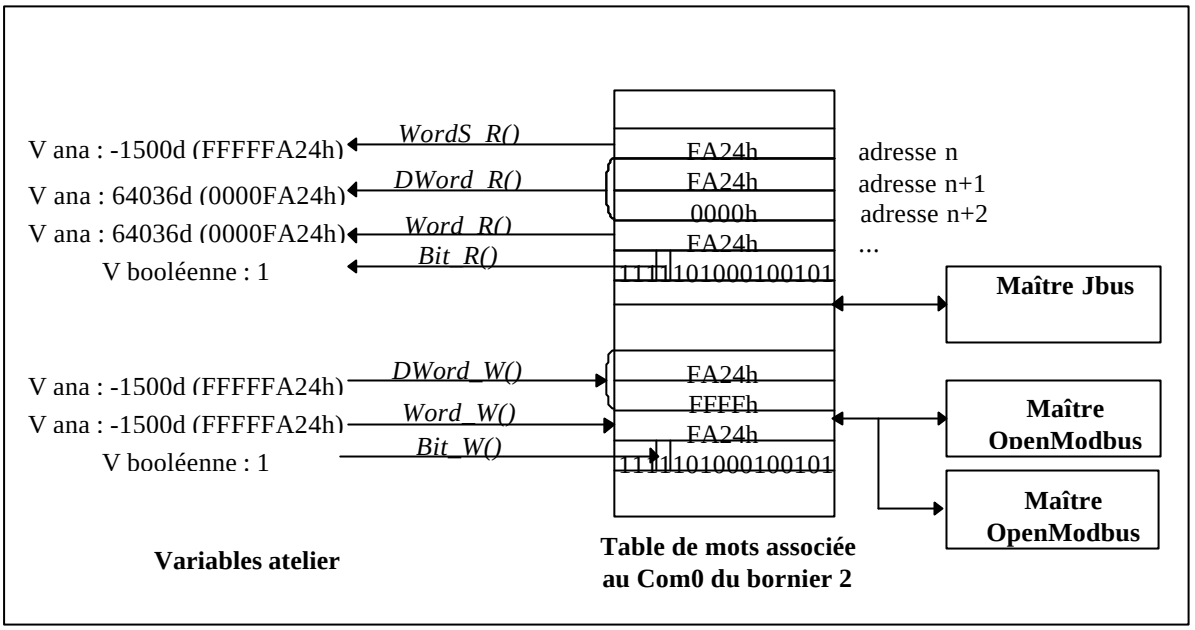


Figure 8 : principe de communication Jbus esclave

Taille et forme des variables entières et des mots contenus dans les table d'échange :

Les variables de type entier sont codées sur 32 bits. Elles peuvent être représentées sous forme

décimale signée [-2147483648..2147483647] ou
hexadécimale non signée [00000000..FFFFFFFF].

Les variables de type mot (16 bits) contenues dans une table d'échange sont lues sous leur forme non signées par la fonction Word_R(). Pour obtenir ces variables sous leur forme signées, il faut utiliser la fonction WordS_R().*Voir les exemples présentés figures 8 et 9.*

Exemple pour un protocole Jbus Maître :

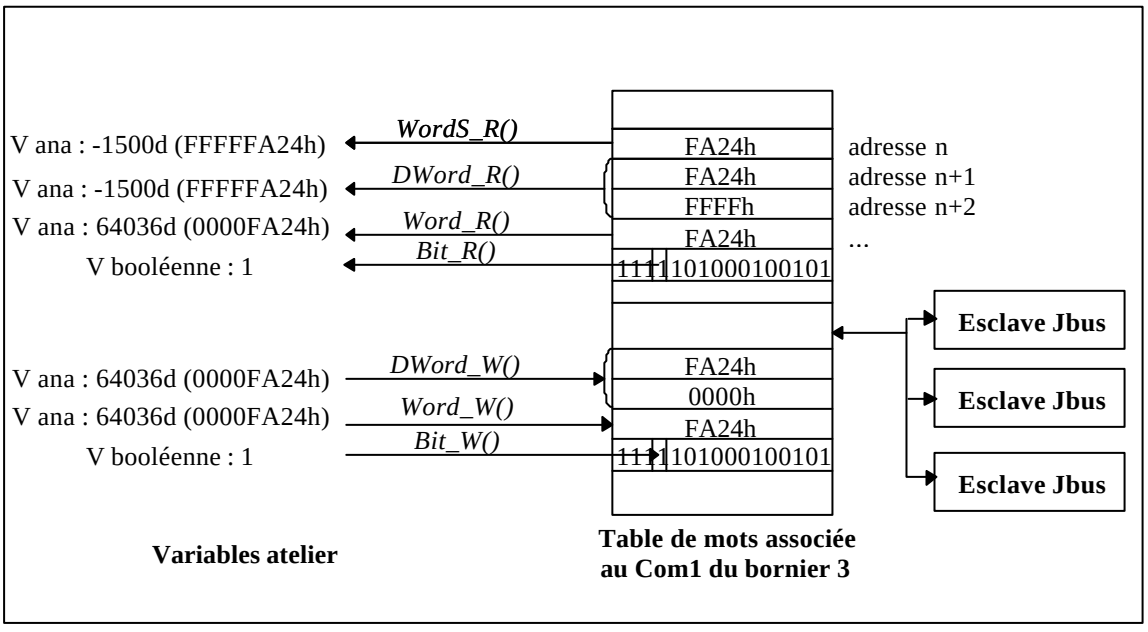


Figure 9 : principe de communication Jbus maître

Les paragraphes suivants détaillent les étapes à suivre afin de mettre en oeuvre chaque protocole.

Important : une table peut être associée à plusieurs ports de communication. Le principe est de déclarer une table par son numéro (de 1 à 7) puis d'utiliser son numéro lors de la déclaration d'un autre port. Si l'on utilise une table déjà déclarée il est obligatoire que la longueur passée en paramètre soit équivalente à la table initiale.

Les ordres Jbus/Modbus reconnus et traités par le LT ISaGRAF sont les suivants :

Fonction	Code Fonction
lecture de plusieurs bits	1 et 2 *(1)*
lecture de plusieurs mots	3 et 4 *(2)*
écriture d'un bit	5
écriture d'un mot	6
écriture de plusieurs mots	16

(1) Le LT ne fait pas la différence entre les bits de sorties et les bits d'entrées

(2) Le LT ne fait pas la différence entre les mots d'entrées et les mots de sorties

Fonction	**Word_R**
ACTION	Lit 1 mot, sous forme non signée, dans une table réseau associée à un port de communication du LT
SYNTAXE	*entier Donnee Word_R(entier NumTable, entier AdrMot);*
PARAMETRES	NumTable : [1..7] AdrMot: [0.."valeur max saisie lors de l'ouverture du Com - 1"]
VALEUR RENVOYEE	Donnée: valeur entière lue [0..FFFFh]. **Les bits de poids fort (2° mot) de la variable entière sont mis à zéro.** Si erreur : 10000h
EXEMPLE	*Donnee := Word_R(1, 2);* (* Lecture d'un mot à l'adresse 2 de la table 1 *)

Fonction	**DWord_R**
ACTION	Lit 2 mots dans une table réseau associée à un port de communication du LT
SYNTAXE	*entier Donnee DWord_R(entier NumTable, entier AdrMot);*
PARAMETRES	NumTable : [1..7] AdrMot: [0.."valeur max saisie lors de l'ouverture du Com - 2"]
VALEUR RENVOYEE	Donnée: valeur entière lue [0..FFFFFFFFh] initialisée à la valeur : -1
EXEMPLE	*Donnee := DWord_R(1, 2);* (* Lecture de 2 mots à l'adresse 2 de la table 1 *)

Fonction	**WordS_R**
ACTION	Lit 1 mot, sous forme signée, dans une table réseau associée à un port de communication du LT
SYNTAXE	*entier Donnee WordS_R(entier NumTable, entier AdrMot);*
PARAMETRES	NumTable : [1..7] AdrMot: [0.."valeur max saisie lors de l'ouverture du Com - 1"]
VALEUR RENVOYEE	Donnée: valeur entière lue [0..FFFFh]. **Les bits de poids fort (2ème) mot de la variable entière sont mis à la valeur 1.** Si erreur : 10000h
EXEMPLE	*Donnee := WordS_R(1, 2);* (* Lecture d'un mot à l'adresse 2 de la table 1 *)

Fonction	**Bit_R**
ACTION	Lit 1 bit dans une table réseau associée à un port de communication du LT
SYNTAXE	*entier Donnee Bit_R(entier NumTable, entier AdrMot, entier RangBit);*
PARAMETRES	NumTable : [1..7] AdrMot: [0.."valeur max saisie lors de l'ouverture du Com - 1"] RangBit: [0..Fh]
VALEUR RENVOYEE	Donnee : valeur entière • bit à 0 : 0 • bit à 1 : 1 • erreur : -1
EXEMPLE	*Donnee := Bit_R(1, 2, 5);* (* lecture du bit 5 du mot 2 de la table 1 *) Conseil : pour avoir un bit en valeur de retour, utiliser la fonction Boo (conversion de donnée entière en type booléen) : Bit := Boo(*Bit_R(1, 2, 5);*

Fonction	**Word_W**
ACTION	Ecrit 1 mot dans une table réseau associée à un port de communication du LT
SYNTAXE	*booleen Status Word_W(entier NumTable, entier AdrMot, entier Donnee);*
PARAMETRES	NumTable : [1..7] AdrMot: [0.."valeur max saisie lors de l'ouverture du Com - 1"] Donnée: valeur entière à écrire [0..FFFFh] **les bits de poids fort (2° mot) ne sont pas pris en compte.**
VALEUR RENVOYEE	Status : [TRUE, FALSE];
EXEMPLE	*Mot := 16#FF;* *Status := Word_W(1, 2, Mot);* (* Ecriture de la variable entière Mot à l'adresse 2 de la table 1 *)

Fonction	DWord_W
ACTION	Ecrit 2 mots dans une table réseau associée à un port de communication du LT
SYNTAXE	*booleen Status DWord_W(entier NumTable, entier AdrMot, entier Donnee);*
PARAMETRES	NumTable : [1..7] AdrMot: [0.."valeur max saisie lors de l'ouverture du Com - 2"] Donnée: valeur entière à écrire [0..FFFFFFFFh]
VALEUR RENVOYEE	Status : [TRUE, FALSE];
EXEMPLE	*Mot := 16#FF008800;* *Status := DWord_W(1, 2, Mot);* (* Ecriture de la variable entière Mot aux adresses 2 et 3 de la table 1 *)

Fonction	Bit_W
ACTION	Ecrit 1 bit dans une table réseau associée à un port de communication du LT
SYNTAXE	*booleen Status Bit_W(entier NumTable, entier AdrMot, entier RangBit, booleen Donnee);*
PARAMETRES	NumTable : [1..7] AdrMot : [0.."valeur max saisie lors de l'ouverture du Com - 1"] RangBit : [0..Fh] Donnée :[TRUE, FALSE]
VALEUR RENVOYEE	Status :[TRUE, FALSE]
EXEMPLE	*EtatBit := TRUE;* *Status := Bit_W(1, 2, 5, EtatBit);* (* bit 5 du mot 2 de la table 1 mis à 1 *)

4.2 Les protocoles sur réseau RS232/RS485

4.2.1 Le protocole Jbus Esclave

Pour utiliser sur un port de communication le protocole Jbus Esclave, 3 fonctions C sont disponibles :

Fonction	**JbusS_O**
ACTION	Ouvre un port Jbus Esclave sur un LT
SYNTAXE	*booleen JbusS_O(entier Bornier, entier Com, entier NumEsclave, entier NumTable, entier LongTable);*
PARAMETRES	Bornier : [1..4] Com : [0,1] NumEsclave : [0..255] NumTable : [1..7] LongTable : [1..4095] 4095 mots (16 bits)
VALEUR RENVOYEE	FALSE : Ouverture non effectuée. TRUE : Ouverture correctement effectuée.
EXEMPLE	*Status := JbusS_O(2, 0, 12, 3, 100);* (* déclare en JbusEsclave le com0 du bornier 2 A ce port est associée une table de 100 mots accessible en lecture/écriture. Cette table est identifiée par le n° 3. Le n° d'esclave est 12. Les paramètres de communication sont par défaut 19200 bauds, sans parité, 1 bit stop, 8 bits de données. Cette table Jbus débute à l'adresse 0 *)

Fonction	**JbusS_C**
ACTION	Ferme un port Jbus Esclave sur un LT
SYNTAXE	*booleen JbusS_C(entier Bornier, entier Com);*
PARAMETRES	Bornier : [1..4] Com : [0,1]
VALEUR RENVOYEE	FALSE : Fermeture non effectuée. TRUE : Fermeture correctement effectuée.
EXEMPLE	*Status := JbusS_C(2, 0);* (* ferme le port Jbus Esclave sur le com0 du bornier 2 *)

Fonction	JbusS_P	
ACTION	Configure les paramètres de communication d'un port Jbus Esclave sur un LT	
SYNTAXE	*booleen JbusS_P(entier Bornier, entier Com, entier Vitesse, entier Parité, entier BitStop, entier Données, entier Accès, entier Silence, entier AddrReseau de base);*	
PARAMETRES	Bornier :	[1..4]
	Com :	[0,1]
	Vitesse :	• 75 Bauds = 1, • 110 Bauds = 2, • 150 Bauds = 3, • 300 Bauds = 4, • 600 Bauds = 5, • 1200 Bauds = 6, • 2400 Bauds = 7, • 4800 Bauds = 8, • 9600 Bauds = 9, • 19200 Bauds = 10, • 38400 Bauds = 11, • 76800 Bauds = 12, (exclusivement sur com1 bornier 1) • 115 kbauds = 13, (exclusivement sur com1 bornier 1) • 200 Bauds = 14. (exclusivement sur com1 bornier 1)
	Parité :	• Pas de parité=0, • Paire=1, • Impaire=2 • Forcée à 0=3, • Forcée à 1=4.
	BitStop :	• 1Stop=1, • 2Stop=2.
	Données :	• 5Bits=5, • 6Bits=6, • 7Bits=7, • 8Bits=8.
	Accès :	• Lecture/Ecriture=0 • Lecture=1 • Ecriture=2 • Aucun=3
	Silence :	[0..7FFFh]
	AddrReseau :	[0..FFFFh] adresse à partir de laquelle le maître aura accès aux données
VALEUR RENVOYEE	FALSE : Configuration non effectuée. TRUE : Configuration correctement effectuée.	
EXEMPLE	*Status := JbusS_P(2,0, 9, 0, 1, 8, 2, 0, 1000);* (* modifie les paramètres Jbus Esclave sur le com0 du bornier 2. Les paramètres de com sont maintenant : 9600 bauds, sans parité, 1 bit stop, 8 bits de données. L'accès est en lecture seule : 2. La table Jbus associée débute à l'adresse 1000. *)	

Exemple

Déclaration d'un port de communication Jbus esclave sur le com0 du bornier 2 dont les paramètres de communication sont :

- Bornier : 2,
- Com : 0,
- n° esclave : 12,
- n° de table : 3
- table de 100 mots associée à ce port,

- vitesse : 9600 bauds,
- parité : impaire,
- 1 bit de stop,
- 8 bits de données,
- accès en lecture seulement.
- temps de silence (3 caractères par défaut) : 0
- adresse réseau Modbus/Jbus : 0

Utilisation dans un projet ISaGRAF : voir l'exemple de projet « Ltjbuss »

- Ouvrir le port de communication : *JbusS_O(2, 0, 12, 3, 100)*,
- Configurer les paramètres de communication : *JbusS_P(2, 0, 9, 0, 1, 8, 2, 0, 0)*,
- Utiliser un maître Jbus pour lire et écrire des données dans cette table,
- Utiliser les fonctions C adéquates pour rafraîchir les variables atelier,
- Fermer le port de communication en terminant le programme : *JbusS_C(2,0)*.

Communication sur la liaison console

Le **liaison console (com1 du bornier 1 par défaut)** supporte le protocole **Modbus esclave d'ISaGRAF**. Ce protocole permet d'accéder aux variables des applications ISaGRAF par leur adresse réseau. Cette adresse réseau est définie dans le dictionnaire de l'atelier.

Seules les variables de type ***Boolean*** ou ***Analog*** sont accessibles. Les fonctions Modbus reconnues par le protocole ISaGRAF sont les suivantes :

1	Lecture n bits
3	Lecture n mots
5	Ecriture 1 bit
6	Ecriture 1 mot
16	Ecriture n mots

Attention : le protocole Modbus ISaGRAF ne gère pas les codes d'erreurs comme "adresse Modbus inconnue".

Les paramètres de communication de cette liaison console sont par défaut :

- n° esclave : 1,
- vitesse : 19200 bauds,
- parité : sans,
- données : 8 bits,
- bits stop : 1.

Ces paramètres peuvent être modifiés en utilisant le paramètre **params_com de la carte cpu3xx** (cf 3.1.2).

4.2.2 Le protocole Jbus Maître

Pour utiliser sur un port de communication le protocole Jbus Maître, 4 fonctions C sont disponibles:

Fonction	**JbusM_O**
ACTION	Ouvre un port Jbus Maître sur un LT
SYNTAXE	*booleen JbusM_O(entier Bornier, entier Com, entier NumTable entier LongTable, entier TimeOut, entier NbEssais);*
PARAMETRES	Bornier : [1..4] Com : [0,1] NumTable : [1..7] LongTable : [1..4095] 4095 mots (16 bits) TimeOut : [0..7FFFh] en millisecondes NbEssais: [0..9] nombre d'essais supplémentaires sur réponse esclave absent
VALEUR RENVOYEE	FALSE : Ouverture non effectuée. TRUE : Ouverture correctement effectuée.
EXEMPLE	*Status := JbusM_O(2, 0, 1, 100, 500, 3);* (* déclare en Jbus Maître le com0 du bornier 2. A ce port est associée une table d'échange de 100 mots ayant pour numéro : 1. Le time-out est de 500ms et le nombre de tentatives sur esclave absent est 3. Les paramètres de com sont par défaut 19200 bauds, sans parité, 1 bit stop, 8 bits de données. *)

Fonction	**JbusM_P**	
ACTION	Configure la communication sur un port Jbus Maître sur un LT	
SYNTAXE	*booleen JbusM_P(entier Bornier, entier Com, entier Vitesse; entier Parité; entier BitStop; entier Données, entier RecTOut, entier BrdTOut, entier BusyRet);*	
PARAMETRES	Bornier :	[1..4]
	Com :	[0,1]
	Vitesse :	• 75 Bauds = 1, • 110 Bauds = 2, • 150 Bauds = 3, • 300 Bauds = 4, • 600 Bauds = 5, • 1200 Bauds = 6, • 2400 Bauds = 7, • 4800 Bauds = 8, • 9600 Bauds = 9, • 19200 Bauds = 10, • 38400 Bauds = 11, • 76800 Bauds = 12, (exclusivement sur com1 bornier 1) • 115 kbauds = 13, (exclusivement sur com1 bornier 1) 200 Bauds = 14. (exclusivement sur com1 bornier 1)
	Parité :	• Pas de parité=0, • Paire=1, • Impaire=2, Forcée à 0=3, Forcée à 1=4
	BitStop :	1Stop=1, 2Stop=2.

	RecTOut :	[0.. 7FFFh] en millisecondes : Temps de silence additionnel aux 3 caractères de détection de fin d'une trame : 0 par défaut
	BrdTOut :	[0.. 7FFFh] en millisecondes : Temps silence additionnel aux 3 caractère de fin d'émission de trame sur diffusion : 0 par défaut
	BusyRet:	[0.. 9] nombre d'essais supplémentaires sur réponse esclave non prêt : 0 par défaut
VALEUR RENVOYEE	FALSE : Configuration non effectuée. TRUE : Configuration correctement effectuée.	
EXEMPLE	*Status := JbusM_P(2, 0, 9, 0, 1, 8, 5, 0, 1);* (* modifie les paramètres Jbus Maître sur le com0 du bornier 2. Les paramètres de com sont maintenant : 9600 bauds, sans parité, 1 bit stop, 8 bits de données. Temps de silence additionnel à la fin de l'émission de la trame : 5ms. 1 tentative supplémentaire sur réponse esclave non prêt *)	

Fonction	**JbusM_T**	
ACTION	2 actions possibles selon la valeur du booléen EnvoiTrame : Envoi d'une trame Jbus Maître sur un port du LT OU lecture du status d'envoi de la trame précédente	
SYNTAXE	*entier JbusM_T(entier Bornier, entier Com, entier NumEsclave, entier CodeFonction, entier AdresseEsclave, entier Longueur, entier AdresseDonnees, booleen EnvoiTrame);*	
PARAMETRES	Bornier :	[1..4]
	Com :	[0,1]
	NunEsclave :	[0..255]
	CodeFonction :	1 ou 2 : lecture n bits ; 3 ou 4 : lecture n mots ; 5 : écriture 1 bit ; 6 : écriture 1 mot ; 15 : écriture n bits ; 16 : écriture n mots
	AdresseEsclave :	[0..FFFFh] adresse modbus du début des données à échanger dans la table de l'esclave.
	Longueur :	[1..128]
	AdresseDonnees :	[0.. "valeur max saisie lors de l'ouverture du Com - 1"] adresse modbus du début des données à échanger dans la table du maître.
	EnvoiTrame :	**TRUE pour envoyer la trame;** **FALSE pour lire le status de la trame envoyée précédemment**
VALEUR RENVOYEE	Si Envoi trame == TRUE : • 0 : Trame non envoyée • 1 : Trame correctement envoyée. Si Envoi trame = FALSE : résultat de la trame envoyée (status au standard JBus) ex : • 0 = échange en cours • 256 = échange correct • 1280 = esclave absent • 770 = adresse incorrecte • 772 = esclave non prêt Voir annexe : liste des codes d'erreurs	
EXEMPLE	(* Sur le com0 du bornier 2 *) (* lecture de 1 mot à l'adresse 5 sur l'esclave 1, rangement à l'adresse 2 *) *Status := JbusM_T(2, 0, 1, 3, 5, 1, 2, TRUE);* (* lecture du status de la trame précédente *)	

	Status := JbusM_T(2, 0, 1, 3, 5, 1, 2, FALSE);

Fonction	**JbusM_C**
ACTION	Ferme un port Jbus Maître sur un LT
SYNTAXE	*booleen JbusM_C(entier Bornier, entier Com);*
PARAMETRES	Bornier : [1..4] Com : [0,1]
VALEUR RENVOYEE	FALSE : Fermeture non effectuée. TRUE : Fermeture correctement effectuée.
EXEMPLE	*Status := JbusM_C(2, 0);* (* ferme le port Jbus Maître sur le com0 du bornier 2 *) Remarque : "StopApplication" du débogueur ferme tous les ports de communication ouverts

Exemple : voir les exemples de projet « Ltjbusm1 » et « Ltjbusm2 »

déclaration d'un port de communication Jbus Maître sur le com0 du bornier 2 dont les paramètres de communication sont :

- Bornier : 2,
- Com : 0,
- n° de table : 1
- table de 100 mots associée à ce port,
- TimeOut : 500 millisecondes.
- NbEssais : 3
- vitesse : 19200 bauds,
- parité : impaire,
- 1 bit de stop,
- 8 bits de données,
- Temps de silence additionnel aux 3 caractères de détection de fin d'une trame : 0,
- Temps silence additionnel aux 3 caractère de fin d'émission de trame sur diffusion : 5,
- Nombre de relance(s) sur réponse esclave non prêt : 1.

Utilisation dans un projet ISaGRAF :

- Ouvrir le port de communication : JbusM_O(2, 0, 1, 100, 500, 3),
- Configurer les paramètres de communication : JbusM_P(2, 0, 9, 0, 1, 8, 5, 0, 1),
- Lire (code fonction 3) 1 mot à l'adresse 5 sur l'esclave n° 6 et le ranger à l'adresse 12 de la table:
- JbusM_T(2, 0, 6, 3, 5, 1, 12, TRUE); envoie la trame de lecture,
- JbusM_T(2 ,0, 6, 3, 5, 1, 12, FALSE); retourne le status de communication.
- Fermer le port de communication en terminant le programme : JbusM_C(2, 0).

4.2.3 Le protocole d'émission / réception d'octets

L'utilisateur du LT ISaGRAF a la possibilité d'implanter un protocole d'émission/réception d'octets sur les liaisons séries disponibles (hormis la liaison console). Les fonctions C fournies permettent d'installer et de gérer des files d'attente de type FIFO : une en émission et une en réception. Une liaison série est gérée indifféremment en RS232 ou en RS485. Ce protocole simple permet de gérer des terminaux, des appareils avec un protocole ASCII sans avoir les contraintes temporelles liées à l'émission et à la réception d'octets. La gestion bas niveau d'un port série est effectuée par le LT sous interruption.

Après une initialisation de la liaison série, l'utilisateur peut lire ou écrire des octets dans les files d'émission et de réception. C'est le LT ISaGRAF qui se charge, sous interruption, d'émettre ou de recevoir les octets sur la ligne.

Fonction	**NulPro_O**
ACTION	Ouvre une communication simple sur un port du LT
SYNTAXE	*booleen NulPro_O(entier Bornier, entier Com, entier LongTabRec, entier LongTabEmi, entier Mode);*
PARAMETRES	Bornier : [1..4] Com : [0,1] LongTabRec : [1..1020] longueur de la file de réception (1020 = 4 messages) LongTabEmi : [1..510] longueur de la file d'émission (510 = 2 messages) Mode : [0..1] mode half-duplex (1) ou Full-Duplex (0)
VALEUR RENVOYEE	FALSE : Ouverture non effectuée. TRUE : Ouverture correctement effectuée.
EXEMPLE	*Status := NulPro_O(2, 0, 1020, 510, 1);* (* déclare une communication simple sur le com0 du bornier 2 A ce port est associée une file de réception de 1020 mots A ce port est associée une file d'émission de 510 mots on est en mode Half Duplex *)

Fonction	**NulPro_S**
ACTION	Ecrit dans la file d'émission sur un port du LT
SYNTAXE	*booleen NulPro_S(entier Bornier, entier Com, entier AddrReseauMsg, entier NbCar);*
PARAMETRES	Bornier : [1..4] Com : [0,1] AddrReseauMsg : [0..FFFFh] adresse réseau du dictionnaire ISaGRAF nota : l'adresse réseau du message doit être déclarée (>0) NbCar : [1..255] 255 : longueur max. d'un message ISaGRAF si NbCar = 0 écriture de tous les caractères du message si NbCar = n écriture de n caractères du message
VALEUR RENVOYEE	FALSE : Ecriture non effectuée. TRUE : Ecriture correctement effectuée.
EXEMPLE	Status := NulPro_S(2, 0, 16#A0, 0); (* envoie le message se trouvant à l'adresse 16#A0 sur la file d'émission associée au com0 du bornier 2 *)

Fonction	**NulPro_R**
ACTION	Lit dans la file de réception sur un port du LT
SYNTAXE	*entier NulPro_R(entier Bornier, entier Com, entier AddrReseauMsg,*

	entier NbCar);
PARAMETRES	Bornier : [1..4] Com : [0,1] AddrReseauMsg : [0..FFFFh] adresse réseau du dictionnaire ISaGRAF nota : l'adresse réseau du message doit être déclarée (>0) NbCar : [1..255] 255 : longueur max. d'un message ISaGRAF si NbCar = 0 lecture de tous les caractères du message si NbCar = n lecture de n caractères du message
VALEUR RENVOYEE	[1..n] : lecture du message correcte 0 : lecture du message non réalisée -1: lecture du message réalisée mais incorrecte : le message ne permet pas de stocker tous les caractères -2: lecture du message réalisée mais incorrecte : un caractère n'a pu être lu dans la file de réception ou la demande de lecture de caractères porte sur un nombre de caractères supérieur à celui du message. -3 : pas de caractères dans le buffer
EXEMPLE	Status := NulPro_R(2, 0, 16#A0, 20); (* lit 20 caractères dans la file de réception associée au com0 du bornier 2 ces caractères sont rangés dans le message se trouvant à l'adresse réseau 16#A0 *)

Fonction	**NulPro_N**
ACTION	Lit le nombre de caractères se trouvant dans la file de réception sur un port du LT
SYNTAXE	*entier NulPro_N(entier Bornier, entier Com);*
PARAMETRES	Bornier : [1..4] Com : [0,1]
VALEUR RENVOYEE	NbCar : nombre de caractères dans la file de réception
EXEMPLE	NbCar := NulPro_N(2, 0); (* lit le nombre de caractères se trouvant dans la file de réception associée au com0 du bornier 2 *)

Fonction	NulPro_P
ACTION	Configure une communication simple sur un port du LT.
SYNTAXE	*booleen NulPro_C(entier Bornier, entier Com, entier Vitesse, entier Parité, entier BitStop, entier Données);*
PARAMETRES	Bornier : [1..4] Com : [0,1] Vitesse : • Vitesse: 75Bauds=1, • 110Bauds=2, • 150Bauds=3, • 300Bauds=4, • 600Bauds=5, • 1200Bauds=6, • 2400Bauds=7, • 4800Bauds=8, • 9600Bauds=9, • 19200Bauds=10, • 38400Bauds=11, • 76800Bauds=12, (exclusivement sur com1 bornier 1) • 115kbauds=13, (exclusivement sur com1 bornier 1) • 200Bauds=14. (exclusivement sur com1 bornier 1) Parité : • Pas de parité=0, • Paire=1, • Impaire=2, • Forcée à 0=3, • Forcée à 1=4.. BitStop : • 1Stop=1, • 2Stop=2. Données : • 5Bits=5, • 6Bits=6, • 7Bits=7, • 8Bits=8.
VALEUR RENVOYEE	FALSE : Configuration non effectuée. TRUE : Configuration correctement effectuée.
EXEMPLE	Status := NulPro_P(2, 0, 9, 0, 1, 8); (* modifie les paramètres de communication sur le com0 du bornier 2 Les paramètres de com sont maintenant : 9600 bauds, sans parité, 1 bit stop, 8 bits de données *)

Fonction	NulPro_C
ACTION	Ferme une communication simple sur un port du LT.
SYNTAXE	*booleen NulPro_C(entier Bornier, entier Com);*
PARAMETRES	Bornier : [1..4] Com : [0,1]
VALEUR RENVOYEE	FALSE : Fermeture non effectuée. TRUE : Fermeture correctement effectuée.
DESCRIPTION	
EXEMPLE	StatusPro_C(2, 0); (* Ferme une communication simple sur le com0 du bornier 2 *)

Exemple

Déclaration d'une communication simple sur un port de communication : com0 du bornier 2. Les paramètres de communication sont :

- Bornier : 2,
- Com : 0,
- file de réception : 1020 mots,
- file d'émission : 510 mots
- Mode : Half Duplex
- vitesse : 19200 bauds,
- parité : impaire,
- 1 bit de stop,
- 8 bits de données,

Utilisation dans un projet ISaGRAF : voir l'exemple de projet « Ltnulpro »

- Ouvrir le port de communication : *NulPro_O(2, 0, 1020, 510, 1);*
- Lire 12 caractères dans la file de réception et les ranger dans le message se trouvant à l'adresse réseau 20h du dictionnaire *NulPro_R(2, 0, 16#20, 12);*
- Ecrire les caractères du message se trouvant à l'adresse réseau 20h du dictionnaire dans la file d'émission : *NulPro_S(2, 0, 16#20, 0);*
- Fermer le port de communication : *NulPro_C(2, 0);*

Exemple : Gestion d'imprimante

Les fonctions d'émission/réception d'octets sur une liaison série permettent de piloter simplement une **imprimante série**. Tout port série du LT peut être employé pour gérer une imprimante série. Les liaisons RS232C permettent en plus de gérer les signaux de contrôle comme DTR ou XON/XOFF.

Seuls les messages possédant une adresse réseau dans le dictionnaire peuvent être imprimés.

Envoi d'un message sur une imprimante connectée au com0 du bornier 2 :

- Ouvrir le port de communication : NulPro_O(2, 0, 1020, 510, 1);
- Configurer si nécessaire les paramètres de communication NulPro_P(...);
- Ecrire les caractères du message se trouvant à l'adresse réseau 20h du dictionnaire NulPro_S(2,0, 16#20, 0);

4.2.4 Signaux de contrôle de la liaison RS232

Hormis la liaison console par défaut (bornier 1, com1), le LT dispose de deux types de liaisons RS232 :

- liaison RS232 comprenant les signaux de contrôle RTS et CTS,
- liaison RS232C comprenant en plus les signaux de contrôle DTR, DSR et DCD.

Ces signaux peuvent être pilotés avec deux fonctions C de lecture ou d'écriture.

Fonction	RS232_R
ACTION	Lit un signal de la liaison RS232
SYNTAXE	*entier Status RS232_R(entier Bornier, entier Com, entier TypeSignal);*
PARAMETRES	Bornier : [1..4] Com : [0, 1] Type Signal : 0=CTS (brin 8) 1=DSR (brin 6) 2=DCD (brin 1)
VALEUR RENVOYEE	Status : 1=état haut (+10V) 0=état bas (-10V) -1=erreur de lecture ou paramètres
EXEMPLE	Status := RS232S_R(2, 0, 1); (* lit le brin DSR sur le com0 du bornier 2 *)

Fonction	RS232_W
ACTION	Pilote (écrit) un signal de la liaison RS232
SYNTAXE	*booleen Status RS232_W(entier Bornier, entier Com, entier TypeSignal, booleen état);*
PARAMETRES	Bornier : [1..4] Com : [0, 1] Type Signal : 0=RTS (brin 7) 1=DTR (brin 4) Etat : TRUE=état haut (+10V) FALSE=état bas (-10V)
VALEUR RENVOYEE	Status : TRUE= signal correctement écrit FALSE= erreur
EXEMPLE	Status := RS232S_W(2, 0, 1, TRUE); (*active le brin RTS sur le com0 du bornier 2*)

Utilisation dans un projet ISaGRAF : voir l'exemple de projet « Lt232sig »

Exemple de lecture du signal DSR :

Status := RS232S_R(2, 0, 1); (* lit le brin DSR sur le com0 du bornier 2 *)

Exemple d'écriture du signal RTS :

Status := RS232S_W(2, 0, 1, TRUE); (* active le brin RTS sur le com0 du bornier 2 *)

Nota : en mode half-duplex le RTS est géré automatiquement (activé à l'émission puis désactivé).

Nota : Les signaux ont un état haut à 10Volts environ et un état bas à -10 Volts environ.

Application : gestion modem

Le modem peut être sélectionné par un signal de la liaison RS232C et la procédure de connexion réalisée avec le protocole simple. Ensuite, les données émises peuvent l'être soit en protocole simple, soit en protocole Modbus/Jbus maître.

4.3 Les protocoles sur réseau ethernet

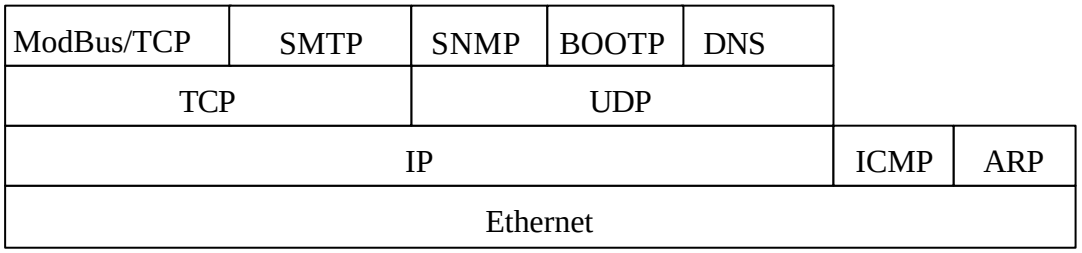
Une fois les paramètres réseau (paragraphe 3.1) correctement saisis , la liaison Ethernet supporte:

Protocoles de la couche liaison d'internet :

- **IP** : Internet Protocol, ensemble de protocoles standards de l'industrie permettant la communication dans un environnement hétérogène : il fournit un protocole de gestion de réseau d'entreprise routable ainsi que l'accès à Internet.
- **ARP** (Address Resolution Protocol) : ARP fournit une correspondance dynamique entre une adresse IP connue et l'adresse matérielle lui correspondant.
- **ICMP** (Internet Control Message Protocol). Le protocole d'interconnexion ICMP permet aux passerelles et aux équipements d'échanger des informations relatives aux conditions anormales.

Protocoles de la couche transport d'internet :

- **TCP :** (Transmission Control Protocol) : TCP procure un service de flux d'octets orienté connexion : les applications dialoguant à travers TCP sont considérées l'une comme un serveur et l'autre comme un client : elles vont établir une connexion avant de pouvoir dialoguer.
- **UDP** : (User Datagram Protocol) : UDP est un mode non connecté : UDP n'utilise pas d'accusé de réception et ne peut donc pas garantir que les données ont bien été reçues : c'est à l'application qui utilise UDP de gérer cet aspect.



Cette suite de protocoles au dessus d'Ethernet détermine le mode de communication des ordinateurs et la procédure de connexion inter-réseaux.

Nota : la fonction ping, du protocole ICMP vous permettra de vérifier la présence d'un équipement sur le réseau en tapant directement son adresse IP ;

exemple d'utilisation : du menu Démarrer de Windows, choisissez la commande exécuter, et taper « ping 192.168.10.1 » pour vérifier la présence du LT connecté, qui porte cette adresse.

Attention : Pour un même projet, dans le cas de l'utilisation simultanée de plusieurs types de protocoles, nous vous conseillons de suivre l'exemple de projet « **ltxmulti** » fourni sur la disquette, et en particulier la méthode d'initialisation séquentielle des différents protocoles, lors de la mise sous tension du LT ethernet.

Identification du LT sur ethernet :

La fonction suivante permet de connaître dans le programme applicatif l'adresse IP du LT :

Fonction	**AddrIP**
ACTION	Retourne l'addresse IP du LT
SYNTAXE	*message Donnée AddrIP();*
PARAMETRES	Aucun
VALEUR RENVOYEE	Adresse IP du LT
EXEMPLE	*AdresseIP = AddrIP();*

4.3.2 Le protocole Modbus/TCP

Ce protocole consiste à encapsuler des échanges Modbus dans des trames IP. Il utilise le mode connecté TCP. Il offre les mêmes fonctionnalités que les voies "modbus" sur les liaisons asynchrones du produit. Les différences avec le protocole Modbus sur voie asynchrone sont les suivantes :

- pas de numéro d'esclave, car l'adressage s'effectue avec l'adresse IP
- utilisation du mode connecté TCP. Ce qui permet la connexion simultanée avec 4 maîtres maximum sur le réseau.
- pas de diffusion possible sur le réseau.

4.3.2.1 Le protocole Modbus/TCP Esclave

Deux fonctions C sont disponibles pour gérer ce protocole.

Fonction	**TCPMbS_O**
ACTION	Ouvre le port Modbus/TCP Esclave sur un LT Ethernet **ATTENTION : le port Modbus/TCP esclave ne peut être ouvert qu'une seule fois.** Si il est ouvert une deuxième fois, le port sera refermé.
SYNTAXE	*booleen tcpmbs_O(entier NumTable, entier LongTable, entier Acces);*
PARAMETRES	NumTable : [1..7]] on affecte une des 7 tables disponibles sur le LT LongTable : [1..4095] 4095 mots (16 bits) Acces : [0..2] 0: Lec/Ecr, 1: Lecture, 2: Ecriture
VALEUR RENVOYEE	FALSE : Ouverture non effectuée. TRUE : Ouverture correctement effectuée.
EXEMPLE	Status = TCPMbS_O(1, 100, 0); (* déclare le protocole Modbus/TCP Esclave sur une CPU LT Ethernet. A ce port est associé la table numéro 1 de 100 mots accessible en lecture/écriture. Cette voie esclave est identifiée par l'adresse IP du LT. Remarque: Quatre maîtres peuvent accéder "simultanément" à cette table.

Fonction	TCPMbS_S
ACTION	Surveillance de la présence de 1 à 4 maîtres sur le port ModBus/TCP esclave
SYNTAXE	*booleen tcpmbs_S(entier NumMaître, message AdresseIP, entier TimeOut);*
PARAMETRES	NumMaître : [1..4] on a au maximum 4 maîtres connectés en même temps (numéro logique associé à l'adresse IP pour permettre de discriminer les différents maîtres AdresseIP : variable de type message contenant l'adresse IP du maître à surveiller TimeOut : [0..FFFF] TimeOut en millisecondes
VALEUR RENVOYEE	FALSE : Maître absent. TRUE : Maître présent.
EXEMPLE	Status = TCPMbS_S (1, adresse_M, 5000); (* surveille la présence du maître dont l'adresse IP est contenue dans la variable « adresse_M » de type message. Ce maître aura pour index 1. Cet index sert à identifier le maître que l'on ne surveillera plus lorsqu'un nouveau maître vient se connecter alors que 4 maîtres étaient déjà en surveillance. Si ce maître est absent au moins 5 secondes, le status passera de TRUE à FALSE *) notes: - une adresse IP a le format xxx.xxx.xxx.xxx avec xxx [0..255]

Exemple de mise en oeuvre dans un projet ISaGRAF : voir le projet « lttcpe »

4.3.2.2 Le protocole Modbus/TCP Maître

Deux fonctions C sont disponibles pour gérer ce protocole :

Fonction	TCPMbM_O
ACTION	Ouvre le port Modbus/TCP Maître sur un LT Ethernet **ATTENTION : le port Modbus/TCP maître ne peut être ouvert que 3 fois (sur trois voies différentes).**
SYNTAXE	*booleen tcpmbm_O(entier NumVoie, entier NumTable, entier LongTable, entier Acces);*
PARAMETRES	NumVoie : [1..3] on a au maximum 3 voies Modbus TCP Maître sur un port Ethernet NumTable : [1..7]] on affecte une des 7 tables disponibles sur le LT LongTable : [1..4095] 4095 mots (16 bits) Acces : [0..2] 0: Lec/Ecr, 1: Lecture, 2: Ecriture
VALEUR RENVOYEE	FALSE : Ouverture non effectuée. TRUE : Ouverture correctement effectuée.
EXEMPLE	Status = TCPMbM_O(2, 1, 100, 0); (* Ouvre une voie (numéro 2) Modbus TCP Maître sur le port Ethernet .A ce port est associée la table numéro 1 de 100 mots accessible en lecture/écriture.

Fonction	TCPMbM_T
ACTION	Envoi d'une trame Modbus TCP Maître sur une voie du port Ethernet du LT et lecture du Status
SYNTAXE	*entier tcpmbm_T((entier NumVoie, entier NumEsclave, entier CodeFonction, entier AddrEsclave, entier Longueur, entier AdresseDonnees);*
PARAMETRES	NumVoie: [1..3] 3 voies mobbus TCP Maître disponible sur le LT NumEsclave : Adresse IP ou adresse symbolique (Adresse DNS)du produit à accéder CodeFonction : 1 ou 2 : lecture n bits 3 ou 4 : lecture n mots 5 : écriture 1 bit 6 : écriture 1 mot 7 : lecture status CPU 15 : écriture n bits 16 : écriture n mots AddrEsclave : [0..0xFFFFh] adresse modbus du début des données à échanger dans la table de l'esclave. Longueur : [1..128] AdresseDonnees : [0..0xFFFFh] adresse modbus du début des données à échanger dans la table du maître.
VALEUR RENVOYEE	Réponse de l'esclave au standard Modbus TCP : Voir annexe 3 : liste des codes d'erreurs
EXEMPLE	Status = TCPMbM_T (2, "192.168.2.4", 3, 5, 1, 2); (* Sur la voie 2, lecture de 1 mot à l'adresse 5 sur l'esclave "192.168.2.4", rangement à l'adresse 2 de la table associée à la voie 2 *) notes: une adresse IP a le format xxx.xxx.xxx.xxx avec xxx [0..255]

Utilisation dans un projet ISaGRAF : voir l'exemple de projet « lttcpm »

4.3.3 Le protocole SNMP V1

SNMP : *Simple Network Management Protocol* : Protocole standard d'administration des hôtes, routeurs et autres appareils sur le réseau IP.

SNMP est basé sur le **modèle Agent/Station de gestion** (ou Agent/Manager). Un agent est un veilleur capable de répondre à des requêtes provenant d'une station de gestion. SNMP utilise UDP comme protocole de transport. Le numéro de port identifiant le protocole application SNMP est 161. Le LT est agent SNMP et les seules opérations supportées sont :

- Get : permet à une station de gestion d'extraire la valeur d'un objet (variable SNMP) d'un agent (LT).
- GetNext : permet à une station de gestion d'extraire la valeur de l'objet suivant (variable SNMP) d'un agent (LT).
- Set : permet à une station de gestion de modifier la valeur d'un objet (variable SNMP) d'un agent (LT).

Les variables SNMP appartenant au LT sont accessibles en lecture/écriture par le manager SNMP : voir le paragraphe **Erreur ! Source du renvoi introuvable.** pour le paramétrage de l'adresse IP du manager.

Les variables gérées par le protocole SNMP appartiennent à une structure unique appelée MIB (Management Information Base) : la MIB est une base de données définie de façon formelle en langage ASN1 (Abstract Syntax Notation 1).

4.3.3.1 MIB II

La MIB II est une MIB standard que tous les agents possèdent. Les différents élements qu'elle comporte sont renseignés ou pas.

L'ID de la MIB II est : 1.3.6.1.2.1…

La MIB II est composée de 10 sous ensembles :

Le LT a uniquement le premier sous-ensemble renseigné : sous ensemble System(1)

System contient les informations de base pour la reconnaissance de l'agent :

Nb : la notation est la suivante : nom-objet(position, type, accès).

- **sysDescr**(1, octet string, read-only): Description de l'agent.
- **SysObjectID**(2, object identifier, read-only) : Identification : Pointe sur la branche du produit (1.3.6.1.4.1.4273 pour LAI)
- **SysUpTime**(3, Time Ticks, read-only) : Temps écoulé (en centièmes de seconde) depuis la réinitialisation.
- **SysContact**(4, octet string, read-write): Personne à contacter.
- **SysName**(5, octet string, read-write) : Nom du nœud.
- **SysLocation**(6, octet string, read-write): Emplacement physique de l'agent.
- **SysServices**(7, integer, read-only): Niveau de services possibles (entre 1 et 7 :couches OSI).

4.3.3.2 MIB LAI

Le protocole SNMP permet d'accéder aux variables du LT définies par ISaGRAF dans la MIB.

Leroy Automation a obtenu auprès de « Internet Assigned Numbers Authority – MIB » une branche identifiée dans la branche Enterprises (4273) : « LAI ».

Dans la branche LAI (4273), une sous branche est définie par le numéro d'agent, identifiant le LT (paramétré dans le câblage de la carte CPU : paramètre « Num_Agent_SNMP » : voir le paragraphe**Erreur ! Source du renvoi introuvable.**), et dans cette sous branche, sont définies les variables ISaGRAF.

2 types de variables ISaGRAF peuvent être utilisés par une station de gestion SNMP: variable de type Entier et variable de type Message.

Pour qu'une station de gestion puisse utiliser une variable du dictionnaire comme variable SNMP et donc réaliser un Get, GetNext ou un Set, il faut ajouter cette variable ISaGRAF à la MIB *(Management Information Base)* SNMP. Pour cela 2 fonctions sont disponibles, respectivement pour chaque type de variable :

Les fonctions **SnmpVA_C(), SnmpVM_C()** permettent de créer respectivement les variables SNMP de types entiers signés sur 32 bits et messages sous forme de chaîne de caractères dont on définit la longueur lors de la création de la variable :

Fonction	**SnmpVA_C**
ACTION	Définit le rang SNMP, dans la branche 1.3.6.1.4.1.4273 d'une variable de type entier du dictionnaire ISaGRAF
SYNTAXE	*booleen Status := SnmpVA_C(entier Flag, entier OID, entier AddrVarAna);*
PARAMETRES	Flag: [1..4] : non utilisé, à 0 par défaut OID: [1..32767] rang de la variable Snmp permettant de l'identifier dans la branche 1.3.6.1.4.1.4273.Num_Agent. AddrVarAna : [0..FFFF] adresse réseau dictionnaire(hexa) de la variable de type entier
VALEUR RENVOYEE	FALSE : Opération non effectuée. TRUE : Opération correctement effectuée.
EXEMPLE	Status := SnmpVA_C(0, 4, 16#20); (* la variable de type entier ayant l'adresse 16#0020 dans le dictionnaire est accessible par SNMP ; son identificateur est le suivant : *1.3.6.1.4.1.4273.2.4.0* *)

Fonction	**SnmpVM_C**
ACTION	Définit le rang SNMP, dans la branche 1.3.6.1.4.1.4273 d'une variable message du dictionnaire ISaGRAF
SYNTAXE	*booleen Status := SnmpVM_C(entier Flag, entier OID, entier AddrVarMess);*
PARAMETRES	Flag: [1..4] : non utilisé, à 0 par défaut OID: [1..32767] rang de la variable Snmp permettant de l'identifier dans la branche 1.3.6.1.4.1.4273.Num_Agent. AddrVarMess : [0..FFFF] adresse réseau dictionnaire(hexa) de la variable message
VALEUR RENVOYEE	FALSE : Opération non effectuée. TRUE : Opération correctement effectuée.
EXEMPLE	Status := SnmpVM_C(0, 14, 16#30); (* la variable message ayant l'adresse 16#0030 dans le dictionnaire est accessible par SNMP ; son identificateur est le suivant : *1.3.6.1.4.1.4273.2.14.0* *)

Chacune de ces fonctions associe l'adresse réseau du dictionnaire, qu'il faut préalablement déclarer, à un index numérique de la variable dans la MIB.

Exemple : voir le projet « LTsnmp »

Adresse de la variable SNMP « entier4 » :

iso.org.dod.internet.private.enterprises.lai.NumeroAgentSNMP.entier4

son identificateur, appelé OID, s'écrit *1.3.6.1.4.1.4273.2.4.0* (0 étant l'instance de la variable portant ce nom).

Enrichissement de la MIB du manager en langage ASN1 :

```
LAI DEFINITIONS ::= BEGIN

        IMPORTS
              enterprises
                    FROM RFC1155-SMI
            OBJECT-TYPE
              FROM RFC-1212;

lai     OBJECT IDENTIFIER ::= { enterprises 4273 }
agent       OBJECT IDENTIFIER ::= { lai 2 }

      entier4     OBJECT-TYPE
           SYNTAX  INTEGER
           ACCESS  read-only
           STATUS  mandatory
           DESCRIPTION
            "test variable entière avec LT Isagraf"
           ::= { agent 4 }

      message4  OBJECT-TYPE
           SYNTAX  OCTET STRING (SIZE (0..255))
           ACCESS  read-only
           STATUS  mandatory
           DESCRIPTION
            "test variable message avec LT Isagraf"
           ::= { agent 14 }

END
```

4.3.3.3 Traps SNMP V1 :

4.3.3.3.1 Trap standard

Un trap de démarrage à froid est émis automatiquement vers le manager à chaque démarrage du produit.

4.3.3.3.2 Traps Spécifiques

Il est également possible d'émettre vers le Manager des traps spécifiques à l'aide de deux fonctions :

- **Trapint** : pour transmettre un code et une valeur.
- **Trapstr** : pour transmettre un code et un message.

Fonction	**Trapint**
ACTION	Envoi un trap avec une valeur associée
SYNTAXE	*booleen Trapint (entier Code, entier Value, entier OID);*
PARAMETRES	Code : valeur du code du trap Value : Entier contenant la valeur à renvoyer OID : [1..32767] rang de la variable Snmp permettant de l'identifier dans la branche 1.3.6.1.4.1.4273.Num_Agent.
VALEUR RENVOYEE	FALSE : Envoi non effectué. TRUE : Envoi correctement effectué
EXEMPLE	*Status := Trapint(3, Entier, 4);* *l'OID associé est donc : 1.3.6.1.4.1.4273.2.4.0*

Fonction	**Trapstr**
ACTION	Envoi un trap avec un message associé
SYNTAXE	*booleen Trapstr (entier Code, message Mess, entier OID);*
PARAMETRES	Code : valeur du code du trap Mess : Message à émettre OID : [1..32767] rang de la variable Snmp permettant de l'identifier dans la branche 1.3.6.1.4.1.4273.Num_Agent.
VALEUR RENVOYEE	FALSE : Envoi non effectué. TRUE : Envoi correctement effectué.
EXEMPLE	Status := Trapstr(7, Mess, *14*); *l'OID associé est donc : 1.3.6.1.4.1.4273.2.14.0*

Exemple : voir le projet « Ltsnmp »

4.3.4 Le protocole SMTP : envoi de courrier électronique

SMTP : *Simple Mail Tranfer Protocol :* Protocole standard sur Internet d'envoi de courriers électroniques

Le nombre de tentatives d'envois de courrier est infini tant que le LT ne parvient pas à atteindre le serveur SMTP. La tentative d'envoi de courrier est avortée si le serveur refuse d'accepter le courrier.

L'objet du courrier électronique envoyé depuis le LT est le suivant : « Message du LT numéro X » avec X=numéro de série du LT.

Les fonctions Email_I() et Email_S() permettent respectivement d'initialiser l'adresse du serveur du courrier sortant (SMTP) et d'envoyer un courrier électronique.

Fonction	**EMail_I()**
ACTION	Initialise l'adresse du serveur SMTP. Cette adresse peut être une adresse IP ou le nom du serveur (ex : smtp.free.fr) **ATTENTION : Le serveur de courrier doit être unique ; par conséquent, il est interdit de l'initialiser plusieurs fois.**
SYNTAXE	*booleen Status := EMail_I(Message Adresse);*
PARAMETRES	Adresse : variable de type message contenant l'adresse du serveur SMTP
VALEUR RENVOYEE	FALSE : Opération non effectuée. TRUE : Opération correctement effectuée.
EXEMPLE	Status := EMail_I(Adresse_Serveur); (* L'adresse du serveur de courrier sortant (SMTP) est initialisée. Sa valeur est celle contenue dans la variable Adresse_Serveur de type message *)

Fonction	**EMail_S()**
ACTION	Envoie un courrier électronique via le serveur SMTP.
SYNTAXE	*booleen Status := EMail_S(Message TO, Message FROM, Message Contenu);*
PARAMETRES	TO : variable de type message contenant l'adresse du destinataire du courrier électronique. Cette adresse peut être sous la forme IP ou littérale (ex : 192.168.0.1 ou nom.prénom@provider.fr) FROM : variable de type message contenant l'adresse de l'expéditeur du courrier électronique. Cette adresse peut être sous la forme IP ou littérale (ex : 192.168.0.2 ou nom.prénom@provider.fr) Contenu : variable de type message contenant le corps du courrier électronique.
VALEUR RENVOYEE	FALSE : Opération non effectuée. TRUE : Opération correctement effectuée.
EXEMPLE	Status := EMail_S(Adresse_Destinataire, Adresse_Expéditeur, Courrier); (* Le message contenu dans la variable Courrier est envoyé depuis Adresse_Expéditeur vers Adresse_Destinataire *)

voir l'exemple de projet « lttemail »

5 Les cartes d'entrées/sorties

Pour chaque carte d'entrées/sorties, une **fiche technique** (menu Aide) est disponible dans l'atelier ISaGRAF.

La **led Flt** de chaque carte d'entrées/sorties est gérée par l'électronique de la carte : si celle-ci n'a pas été rafraîchie **au bout de 1 seconde, un monostable pilote cette sortie**. Ce monostable est activé tant que la carte n'a pas été initialisée.

Nota :ce monostable est égal à 100 millisecondes sur les DIO130 et 200 millisecondes sur les DI130.

5.1 Carte DI310

La carte DI310 est composée de 32 entrées de type booléen.

5.2 Carte DI410

La carte DI410 est composée de 64 entrées de type booléen.

5.3 Carte DO310

La carte DO310 est composée de 32 sorties de type booléen.

5.4 Carte AI110

La carte AI110 est composée de 8 entrées de type entier (16 bits signés).

5.5 Carte AO121

La carte AO121 est composée de 8 sorties de type entier (16 bits signés).

5.6 Carte AI210

La carte AI210 est composée de 16 entrées de type entier (16 bits signés).

Table de conversion courant/tension <=> nombre de points	
Entrées courant non isolées	±21,1mA => ±32767 points.
Entrées tension non isolées	±10,25V => ±32767 points.
Entrées tension isolées	0 à 10,25V => 32767 points.
Entrées courant isolées	0 à 21,1mA => 32767 points.
Sorties courant	0 / 32767 points => 4 / 20mA
Sorties tension	±32767 points => ±10V

5.7 Equipement DI312

L'équipement DI312 est composé de 2 cartes :

- Inputs: 32 entrées de type booléen
- Fault : 32 défauts associés aux entrées

Les **modules DI312** ont un dispositif paramétrable de comparaison pour **contrôler la filerie** des capteurs en leurs connectant un réseau de 2 résistances : **entrées de sécurité**. Les réseaux de résistances se ramènent à 2 types : le **montage série** (les 2 résistances en série) et le **montage parallèle** (les 2 résistances en parallèle).La résistance série est toujours présente. Dans le montage parallèle, le capteur est en série avec Rp qu'il supprime par ouverture. Dans le montage série, le capteur est en parallèle avec Rp qu'il supprime par fermeture.

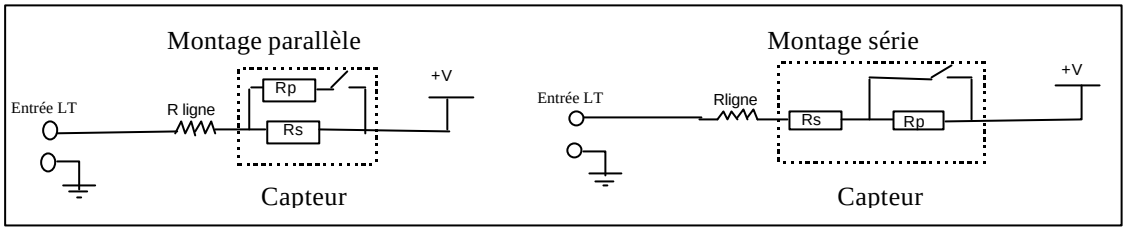


Figure 10 : Câblage des entrées de sécurité

Pour garder la généralité du paramétrage, ISaGRAF propose d'indiquer la résistance équivalente du réseau de résistances lorsque le capteur est **normalement ouvert (Rcno)** et lorsque le capteur est **normalement fermé (Rcnf**).Les valeurs des résistances sont **en OHMS**.

Montage parallèle	Montage série
Rcnf = Rs//Rp + Rligne	Rcnf = Rs + Rligne
Rcno = Rs + Rligne	Rcno = Rs+Rp + Rligne

ATTENTION : Le **paramétrage est unique** pour les valeurs de résistance d'une **carte DI312** donc le même pour toutes les voies d'un même module DI312.

Les paramètres de la carte DI312 sont les suivants :

- Masque de 32 bits du contrôle filerie des 32 entrées. Contrôle de filerie actif sur l'entrée n si le bit de rang n est à l'état 1. Par défaut, les 32 bits du masque sont à l'état 1.
- RCNO : valeur unique pour toutes les entrées.
- RCNF : valeur unique pour toutes les entrées.
- Rligne : valeur unique pour toutes les entrées.

Pour chaque voie, le bit d'état et le bit d'alarme codent 4 états possibles :

Etat Entrée LED verte	Alarme (LED rouge)	Description
0 (éteinte)	0 (éteinte)	capteur normalement ouvert
1 (allumée)	0 (éteinte)	capteur normalement fermé
0 (éteinte)	1 (allumée)	entrée non connectée ou court-circuit au 0V
1 (allumée)	1 (allumée)	court circuit au +V

Contraintes sur les résistances :

- - Les résistances doivent avoir une tolérance de 1% max.
- - 0.7 kΩ < **Rcno** < 22 kΩ
- - Rcno induit le courant dans le dispositif de mesure : I (mA) = 22 (V) / (1+ Rcno (kΩ)) sachant que I doit être compris entre 1 mA et 9.96 mA . Si I calculé est supérieur à 9.96 mA, le saturer à 9.96 mA.
- - (2.95(V) / I (mA)) < **Rcnf** (kΩ) < Rcno (kΩ) - Rligne (kΩ) - 1.95 (V) / I (mA)
- - **Rligne** < 0.2 kΩ

5.8 Equipement DIO210

L'équipement DIO210 est composée de 2 cartes :

- 16 entrées de type booléen
- 8 sorties de type booléen

5.9 Equipement AIO320

L'équipement AIO320 est composée de 3 cartes :

- 8 entrées de type entier
- 16 entrées de type booléen (les entrées [1..8] et [9..16] sont respectivement dédiées aux dépassements du seuil haut et au dépassement du seuil bas des 8 entrées analogiques)
- 4 sorties de type entier

Table de conversion courant / tension <=> nombre de points sur AIO320	
Entrées courant non isolées	±20mA => ±32767 points.
Entrées tension non isolées	±10V => ±32767 points.
Sorties courant	0 / 32767 points => 4 / 20mA
Sorties tension	±32767 points => ±10V
Entrées sondes PT100	-50°C=>-500 points +350°C=>+3500 points

5.10 Equipement DI130

L'équipement DIO210 est composée de 2 cartes :

- 16 entrées de type booléen
- 16 entrées inverses de type booléen

5.11 Equipement DIO130

La carte DIO130 est composée de composée de 3 cartes :

- 8 entrées de type booléen
- 16 sorties de type booléen pour commander 8 relais à commande double.
- 4 sorties de type booléen pour commander les led en face avant

6 Diagnostic et dépannage

6.1 Lecture du status des cartes CPU et d'entrées/sorties

Fonction	IOStatus
ACTION	Lit le status des cartes CPU et d'E/S.
SYNTAXE	*entier Status IOStatus(entier RangCarte);*
PARAMETRES	RangCarte [0..15] avec 0=CPU, 1=1°carte d'ES ... , 15=15° carted'E/S
VALEUR RENVOYEE	Donnée: valeur entière lue [0..FFFFh]
DESCRIPTION	Seuls les status de cartes correctement initialisées sont remontés Status = 0 : carte paramétrée dans l'atelier mais inaccessible sur le bus d'E/S Status = -1 : RangCarte erroné ou carte non paramétrée dans l'atelier Composition du Status : cf. doc "LT ISaGRAF" chapitre III
EXEMPLE	*Status := IOStatus(1);* (* lecture du status de la première carte d'E/S *)

Signification des bits du status de la carte CPU :

• **bit 0**	à 1 a chaque fin de cycle
• **bit 1**	à 1 si Led WDG allumée
• **bit 2**	à 0 si défaut I/O (led I/O allumée)
• **bit 3**	à 1 si initialisations terminées (à 1 à chaque fin de cycle)
• **bit 4**	à 0 si erreur Test (led Test allumée)
• **bit 5**	à 1 si PRM détecté
• **bits 6-15**	Réservé

Signification des bits du status des cartes d'entrées/sorties :

• **bits 0-7**	**Code de la carte [0..FFh] bit 6 non significatif**
• **bit 8**	à 1 si alimentation interne de la carte correcte
• **bit 9**	cf. tableau ci-dessous
• **bit 10**	WDG : à 0 si WDG à l'état 1 (led allumée)
• **bit 11**	cf. tableau ci-dessous
• **bits 12-15**	**Position de la carte sur le bus [0..15]**

Code des cartes d'entrées/sorties :

Carte	Status Bit 11	Status Bit 9	Code de la carte [0..FFh] Masquer le bit 6 : BFh
DI310	1	Al Ext	03h ou 43h
DI312	NS	Al Ext	14h
DI410	1	Al Ext	06h
DO310	Fault	Al Ext	05h ou 45h
DIO210	Monostable	VRel	16h
AI110	0	0	80h
AI210	0	0	81h
AO121	0	0	88h
AIO320	Monostable	0	83h
réservé			30h à 37h
DIO130	Monostable	VRel	58h
DI130	Monostable	0	59h

Avec :

- NS : non significatif
- Al Ext : à 1 si alimentation extérieure présente sur les borniers et comprise entre Valim ±20%.
- (*) Al ext de la DI312 : Alimentation extérieure entre 24V±10%
- Fault : à 0 si surcharge sur une voie de sortie TOR.
- Monostable : à 1 si carte correctement rafraîchie; pour la DIO130 le fonctionnement est inversé.
- VRel : à 1 si les relais sont correctement alimentés.

Fonction	**IOInits**
ACTION	Lit le compteur d'initialisations des cartes d'E/S
SYNTAXE	*entier Status IOInits(entier RangCarte);*
PARAMETRES	RangCarte [1..15] avec 1=1°carte d'ES ... , 15=15° carted'E/S
VALEUR RENVOYEE	Donnée: valeur entière lue [0..FFFFh]
DESCRIPTION	Seuls les compteurs de cartes correctement initialisées sont remontés Le compteur d'init vaut 1 lors d'une initialisation correcte (unique) de la carte.
EXEMPLE	*Status := IOInits(1);* (* lecture du compteur d'initialisations de la première carte d'E/S *)

6.2 Erreurs remontées à l'atelier

Les erreurs remontées à l'atelier en mode débug sont de deux sortes :

- **erreurs codées par ALTERSYS** : texte expliquant l'erreur avec un numéro allant de 0 à 99. Elles sont détaillées dans le guide de l'utilisateur de l'atelier.
- **erreurs codées par LAI** : numéro allant de 100 à 255.

Les erreur codées par LAI sont :

- 100 : type de flash (non AMD512K Bottom). La taille du code TIC est alors limitée à 128Ko.
- 101 à 105 : erreur durant la sauvegarde de l'application TIC en FLASH.
  - 101 : erreur de lecture dans la FLASH.
  - 102 : erreur d'écriture dans la FLASH des infos sur les espaces réservés par une application.
  - 103 : erreur d'accès à 1 secteur de la FLASH.
  - 104 : erreur d'effacement d'un secteur de la FLASH.
  - 105 : erreur de lecture des infos sur les espaces réservés par une application.
- 110 à 112 : erreur durant la lecture de l'application TIC en FLASH.
  - 110 : erreur de lecture du type de FLASH.
  - 111 : erreur d'allocation mémoire pour les espaces réservés par une application.
  - 112 : erreur d'accès à 1 secteur de la FLASH.
- 113 : erreur sur le checksum de l'application TIC lue en FLASH.
- 120 et 121: mémoire dynamique du LT altérée, reboot automatique du LT.
- 130 : insertion de carte interdite dans le cas d'un LT80 (rang[1..3]).
- 140 à 143 : erreur dans la sauvegarde de variables non volatiles :
  - 140 : erreur : si 1 ou plusieurs variables non volatiles sont cochées, il est nécessaire d'en avoir au minimum une de chaque type.
  - 141 : erreur d'allocation mémoire LT.
  - 142 : mémoire secourue pleine.
  - 143 : erreur lecture mémoire secourue.
- 150 : erreur initialisation horloge secourue.
- 151 : erreur écriture horloge secourue.
- 160 : erreur lecture signaux de contrôle RS232.
- 161 : erreur écriture signaux de contrôle RS232.
- 170 : erreur de paramétrage de la communication du com1 du bornier 1 : paramètre params_com de la carte cpu3xx.

6.3 Dépannage de la liaison console : passage du LT en mode paramétrage

Le mode paramétrage est à utiliser pour rétablir le dialogue entre le PC et le LT ISaGRAF, dans le cas de perte de celle-ci suite au changement des paramètres de la liaison série dédiée au dialogue sur le LT ISaGRAF.

Pour un LT ISaGRAF, passer en mode paramétrage, consiste à n'exécuter que le noyau ISaGRAF sans application TIC téléchargée (Target Independant Code).

Ce mode appelé **PRM** est symbolisé par la led du même nom sur la CPU.

Pour passer en PRM :

- mettre le LT hors tension,
- relier le LT (Bornier 1, com1) et le PC (com1 ou com2) avec un câble RS232,

- exécuter le logiciel "**SSTB**" fourni sur la disquette "Librairies". Ce logiciel attend de reconnaître un LT.
- Choisir le port de communication du PC [1..4],
- Choisir le bouton "**PRM**",
- mettre le LT sous tension,
- dans sa séquence d'initialisation le LT passe en mode PRM et allume sa led PRM en fixe. Le PC affiche "**LT mis en paramètrage par défaut**"
    - Le LT ISaGRAF est alors en mode PRM.

Seule la led PRM allumée en fixe garantit un passage réussi en mode PRM.

6.4 Les leds du LT ISaGRAF

6.4.1 Led de l'alimentation

L'alimentation comporte une led verte allumée en fixe si la tension est présente et correcte.

6.4.2 Leds de la CPU

- **LED RUN verte** :
  - ◊ clignote lentement (1s) si l'application TIC (ISaGRAF) est exécutée correctement.
  - ◊ clignote rapidement (1/10s) si on est en PRM, ou l'application est arrêtée par ISaGRAF --> le noyau tourne mais n'exécute pas d'application TIC.
- **LED TEST rouge** :
  - ◊ éteinte si fonctionnement correct.
  - ◊ allumée fixe si le programme lu dans la Flash n'est pas correct ou insertion carte d'E/S non correcte.
- **LED I/O rouge** :
  - ◊ éteinte si fonctionnement correct.
  - ◊ allumée fixe si insertion carte incorrecte ou si un status carte d'E/S au moins est incorrect au cours de l'exécution du programme.
- **LED PRM verte** :
  - ◊ allumée fixe si mode PRM au boot du LT. Ne s'éteint qu'au prochain reboot du LT sans PRM.
  - ◊ éteinte sinon.
- **LED PRG verte** : non gérée par ISaGRAF.
  - ◊ allumée fixe si mode PRG au boot du LT : pont entre les broches 5 et 6 du com 1.1. Ne s'éteint qu'au prochain reboot du LT sans PRG.
  - ◊ éteinte sinon
- **LED WDG rouge** :
  - ◊ allumée fixe par défaut.
  - ◊ éteinte dès que le noyau ISaGRAF tourne ET que le programme lu dans la flash est correct. Ceci est vrai si Le WDG est géré automatiquement par le noyau (cf III.5).
- **LEDs comx verte** :
  - ◊ la led comx de la liaison console est allumée fixe.
  - ◊ autres leds : gestion laissée libre à l'utilisateur. Ex : allumée fixe dès qu'un com est correctement initialisé.

L'utilisateur ne peut avoir d'action que sur les Leds comx et depuis l'atelier ISaGRAF par la fonction ***LedCpu(RangLed, EtatLed)*** :

Fonction	**LedCpu**
ACTION	Pilote les Leds comx de la CPU
SYNTAXE	*booléen LedCPU (entier RangLed, booléen Etat);*
PARAMETRES	RangLed : de 6,8 à 14 Etat : FALSE éteinte, TRUE allumée.
VALEUR RENVOYEE	FALSE : Led non pilotée. TRUE : OK.
DESCRIPTION	Seules les leds 6,8 à 14 peuvent être pilotées.

NOTES	Rang des leds [6,8..14] : • 0 : Led Run : inaccessible • 1 : Led Test : inaccessible • 2 : Led I/O : inaccessible • 3 : Led Prm : inaccessible • 4 : Led Prg : inaccessible • 5 : Led Wdg: inaccessible • 6 : Led com 4.1 • 7 : inutilisé • ---------- • 8 : Led com 1.0 • 9 : Led com 1.1 • 10 : Led com 2.0 • 11 : Led com 2.1 • 12 : Led com 3.0 • 13 : Led com 3.1 • 14 : Led com 4.0 • 15 : inutilisé Un "Stop Application" du deboggeur ISaGRAF n'éteint pas les leds des coms gérées par l'utilisateur (6, 8..14).
EXEMPLE	*Status := LedCPU(9, TRUE);* (* allume la led du com1.1 sur la CPU *)

6.4.3 Pilotage des leds des cartes d'entrées/sorties

Les leds correspondant aux entrées/sorties sont rafraîchies automatiquement par le noyau. Cependant des fonctions sont proposées dans l'atelier pour piloter de manière différente l'état de ces leds (ex : inverser la logique des E/S TOR). Il existe une fonction par carte :

- *LedDI310(RangCarte, Leds_1_32);*
- *LedDI410(RangCarte, Leds_1_32, Leds_33_64);*
- *LedDI312(RangCarte, Leds_1_32, Leds_33_64);*
- *LedDO310(RangCarte, Leds_1_32);*
- *LedDIO210(RangCarte, LedsI_1_16, LedsO_1_8);*
- *LedAI110(RangCarte, LedsV_1_8, LedsR_1_8);*
- *LedAI210(RangCarte, LedsV_1_16, LedsR_1_16);*
- *LedAO121(RangCarte, Leds_1_8);*
- *LedAIO320(RangCarte, Leds_1_8);*

Attention : il faut réécrire la commande des leds à chaque cycle sinon elle est écrasée par le noyau du LT : commande des sorties TOR.

Fonction	**LedDI310**
ACTION	Pilote les Leds d'une carte DI310
SYNTAXE	*booleen LedDI310(entier RangCarte, entier Leds_1_32)*
PARAMETRES	RangCarte : de 1 à 15. Leds_1_32 : état des 32 premières leds (0 éteinte, 1 allumée).
VALEUR RENVOYEE	FALSE : Rafraîchissement non effectué. TRUE : Rafraîchissement correctement effectué.
DESCRIPTION	L'état de chaque led est représenté par un bit.
EXEMPLE	*Status := LedDI310(1, 16#FFFFFFFF);* (* allume toutes les leds d'une DI310 à l'emplacement 1 *)

Fonction	**LedDI410**
ACTION	Pilote les Leds d'une carte DI410
SYNTAXE	*booleen LedDI410(entier RangCarte, entier Leds_1_32; entier Leds_33_64)*
PARAMETRES	RangCarte : de 1 à 15. Leds_1_32 : état des 32 premières leds (0 éteinte, 1 allumée). Leds_33_64 : état des 32 dernières leds (0 éteinte, 1 allumée).
VALEUR RENVOYEE	FALSE : Rafraîchissement non effectué. TRUE : Rafraîchissement correctement effectué.
DESCRIPTION	L'état de chaque led est représenté par un bit.
EXEMPLE	*Status := LedDI410(1, 16#FFFFFFFF, 16#FFFFFFFF);* (* allume toutes les leds d'une DI410 à l'emplacement 1 *)

Fonction	**LedDI312**
ACTION	Pilote les Leds d'une carte DI312
SYNTAXE	*booleen LedDI312(entier RangCarte, entier Leds_1_32; entier Leds_33_64)*
PARAMETRES	RangCarte : de 1 à 15. Leds_1_32 : état des 32 leds vertes (0 éteinte, 1 allumée). Leds_33_64 : état des 32 leds rouges (0 éteinte, 1 allumée).
VALEUR RENVOYEE	FALSE : Rafraîchissement non effectué. TRUE : Rafraîchissement correctement effectué.
DESCRIPTION	L'état de chaque led est représenté par un bit.
EXEMPLE	*Status := LedDI312(1, 16#FFFFFFFF, 16#FFFFFFFF);* (* allume toutes les leds d'une DI312 à l'emplacement 1 *)

Fonction	**LedDO310**
ACTION	Pilote les Leds d'une carte DO310
SYNTAXE	*booleen LedDO310(entier RangCarte, entier Leds_1_32)*
PARAMETRES	RangCarte : de 1 à 15 Leds_1_32 : état des 32 leds rouges (0 éteinte, 1 allumée).
VALEUR RENVOYEE	FALSE : Rafraîchissement non effectué. TRUE : Rafraîchissement correctement effectué.
DESCRIPTION	L'état de chaque led est représenté par un bit.
EXEMPLE	*Status := LedDO310(1, 16#FFFFFFFF);* (* allume toutes les leds d'une DO310 à l'emplacement 1 *)

Fonction	**LedDIO21**
ACTION	Pilote les Leds d'une carte DIO210
SYNTAXE	*booleen LedDIO21(entier RangCarte, entier Ledsi_1_16, entier Ledso_1_8)*
PARAMETRES	RangCarte : de 1 à 15 Ledsi_1_16 : état des 16 leds vertes (0 éteinte, 1 allumée). Ledso_1_8 : état des 8 leds rouges (0 éteinte, 1 allumée).
VALEUR RENVOYEE	FALSE : Rafraîchissement non effectué. TRUE : Rafraîchissement correctement effectué.
DESCRIPTION	L'état de chaque led est représenté par un bit.
EXEMPLE	*Status := LedDIO21(1, 16#FFFF, 16#FF);* (* allume toutes les leds d'une DIO210 à l'emplacement 1 *)

Fonction	**LedAI110**
ACTION	Pilote les Leds d'une carte AI110
SYNTAXE	*booleen LedAI110(entier RangCarte, entier LedsV_1_8; entier LedsR_1_8)*

PARAMETRES	RangCarte : de 1 à 15. LedsV_1_8 : état des 8 leds vertes (0 éteinte, 1 allumée). LedsR_1_8 : état des 8 leds rouges (0 éteinte, 1 allumée).
VALEUR RENVOYEE	FALSE : Rafraîchissement non effectué. TRUE : Rafraîchissement correctement effectué.
DESCRIPTION	L'état de chaque led est représenté par un bit.
EXEMPLE	*Status := LedAI110(1, 16#FF, 16#FF);* (* allume toutes les leds d'une AI110 à l'emplacement 1 *)

Fonction	**LedAI210**
ACTION	Pilote les Leds d'une carte AI210
SYNTAXE	*booleen LedAI210(entier RangCarte, entier LedsV_1_16; entier LedsR_1_16)*
PARAMETRES	RangCarte : de 1 à 15. LedsV_1_16 : état des 16 leds vertes (0 éteinte, 1 allumée). LedsR_1_16 : état des 16 leds rouges (0 éteinte, 1 allumée).
VALEUR RENVOYEE	FALSE : Rafraîchissement non effectué. TRUE : Rafraîchissement correctement effectué.
DESCRIPTION	L'état de chaque led est représenté par un bit.
EXEMPLE	*Status := LedAI210(1, 16#FFFF, 16#FFFF);* (* allume toutes les leds d'une AI210 à l'emplacement 1 *)

Fonction	**LedAO121**
ACTION	Pilote les Leds d'une carte AO121
SYNTAXE	*booleen LedAO121(entier RangCarte, entier Leds_1_8)*
PARAMETRES	RangCarte : de 1 à 15. Leds_1_8 : état des 8 leds vertes (0 éteinte, 1 allumée).
VALEUR RENVOYEE	FALSE : Rafraîchissement non effectué. TRUE : Rafraîchissement correctement effectué.
DESCRIPTION	L'état de chaque led est représenté par un bit.
EXEMPLE	*Status := LedAO121(1, 16#FF);* (* allume toutes les leds d'une AO121 à l'emplacement 1 *)

Fonction	**LedAIO320**
ACTION	Pilote les Leds d'une carte AIO320
SYNTAXE	*booleen LedAIO32(entier RangCarte, entier Leds_1_8)*
PARAMETRES	RangCarte : de 1 à 15. Leds_1_8 : état des 8 leds oranges (0 éteinte, 1 allumée).
VALEUR RENVOYEE	FALSE : Rafraîchissement non effectué. TRUE : Rafraîchissement correctement effectué.
DESCRIPTION	L'état de chaque led est représenté par un bit.
EXEMPLE	*Status := LedAIO32(1, 16#FF);* (* allume toutes les leds d'une AIO320 à l'emplacement 1 *)

6.5 Identification du LT

6.5.1 Version du noyau embarqué sur la cible LT

Fonction	LTVer
ACTION	Lit la version du noyau embarqué sur la cible
SYNTAXE	*entier LTVer();*
PARAMETRES	Aucun
VALEUR RENVOYEE	version 1.04 : 0104h PF = 1 et pf =4 version 1.1 : 0110h PF = 1 et pf =10h
EXEMPLE	*VersionLT := LTVer();*

6.5.2 Type de CPU montée sur le LT

Fonction	LTType
ACTION	Lit le type de CPU montée sur le LT et le type de carte fille
SYNTAXE	*entier TypeLT := LTType();*
PARAMETRES	Aucun
VALEUR RENVOYEE	**Octet de poids faible : type de LT** 2,3 : LT160 4,5 : ACS21 6,7 : LT80 autre : cpu non identifiée **Octet de poids fort : type de carte fille** 13 : 1 seul bornier de communication : Com301 11 : Com301 + 1 bornier de communication 7 : Com301 + 2 à 3 borniers de communication 14 : Com303
EXEMPLE	*TypeLT := LTType();*

6.5.3 Numéro de série du LT

Fonction	LTSerial
ACTION	Lit le numéro de série du LT
SYNTAXE	*entier Donnée LTSerial();*
PARAMETRES	Aucun
VALEUR RENVOYEE	Numéro de série : [0..FFFFh]
EXEMPLE	*NumSerie := LTSerial();*

ANNEXE : Liste des codes d'erreur (status de communication) Modbus/Jbus asynchrone et modbus/TCP maître

Décimal	Hexa	Commentaire
0	**0**	**échange en cours**
256	**100**	**échange correct**
769	301	code d'exception : fonction inconnue
770	302	code d'exception : adresse incorrecte
771	303	code d'exception : donnée invalide
772	304	code d'exception : esclave non prêt
773	305	code d'exception : acquittement
774	306	code d'exception : non acquittement
775	307	code d'exception : erreur d'écriture
776	308	code d'exception : chevauchement de zones
896	380	erreur de connexion
897	381	avertissement de connexion
1024	400	numéro d'esclave incorrect
1025	401	code fonction incorrect
1026	402	longueur incorrecte
1027	403	code sous-fonction incorrect
1028	**404**	**adresse incorrecte**
1029	405	donnée incorrecte
1030	406	longueur de trame incorrecte
1280	**500**	**esclave absent**
1281	501	erreur de CRC
4096	1000	à l'émission : trame en cours
4097	1001	à l'émission : erreur de diffusion
4099	1004	à l'émission : longueur erronée
4100	1005	à l'émission : erreur d'offset
4101	1006	à l'émission : erreur de fonction
4102	1007	à l'émission : erreur de sous-fonction
4103	1008	à l'émission : erreur de donnée sous-fonction
4104	1009	à l'émission : erreur de stockage

en gras les codes d'erreur, les status de communication les plus fréquemment rencontrés.

TABLE DES FIGURES

INDEX DES FONCTIONS C