Modèle CEI 61131

Table des matières

Introduction à la norme CEI 61131

Ce chapitre du manuel décrit la norme CEI 61131, d'après laquelle les produits d'automatisation SELECONTROL® MAS sont effectivement structurés au moyen de l'outil de programmation CAP 1131.

Pourquoi une norme ?

Tandis que le programmeur de langages évolués s'appuyait depuis longtemps sur un environnement de programmation normalisé, le programmeur d'automates programmables devait réapprendre une programmation spécifique pour chaque nouveau système d'API. Non seulement le langage de programmation différait, mais également les possibilités de structurer un programme. Ainsi, la transposition de programmes existants d'un système à l'autre demandait des efforts considérables.

La complexité sans cesse croissante des applications à base d'automates programmables a fait naître la nécessité d'une harmonisation, autrement dit la normalisation de la programmation des automates programmables.

La norme CEI 61131 est une norme reconnue sur le plan international, servant de base pour pratiquement tous les nouveaux développements en matière d'automation. De façon délibérée, le champ d'application de la norme a été grandement étendu afin de pouvoir regrouper également les projets à base d'ordinateurs industriels ainsi que les combinaisons de divers domaines d'application, tels que les techniques de commande et d'entraînement. Les modules en langage évolué ont ainsi été harmonisés avec les principes de la technique de programmation des automates programmables.

Avantages de la norme CEI 61131 :

- organisation uniforme des programmes
- langages de programmation normalisés
- types de données définis
- concept de variables avec encapsulage des données
- bibliothèques de fonctions normalisées

Note :

La norme exige de tous les produits s'appuyant sur CEI 61131, une soi-disant « liste de compatibilité » dans laquelle figurent sous forme tabellaire toutes les fonctionnalités normalisées. Pour tout produit CEI 61131, il doit être indiqué dans la « liste de compatibilité » les fonctionnalités pouvant être satisfaites ou non par ce produit.

Cette liste figure en annexe de ce présent Manuel du système.

Organisation de la norme

La norme est divisée en cinq parties :

Partie 1	**Aperçu général et définitions** Elaboration de définitions de base s'appliquant dans le monde de l'automatisation, afin que les utilisateurs et les fabricants parlent le même langage (par exemple, définition du langage LI).
Partie 2	**Matériel (signaux d'E/S, caractéristiques liées à la sécurité, environnement)** Définition des caractéristiques électriques, mécaniques et fonctionnelles d'une solution d'automatisation, définition des informations demandées par le fabricant et des méthodes de test à appliquer.
Partie 3	**Langages de programmation** Cette partie définit le modèle logiciel, y compris la syntaxe et la sémantique des langages de programmation ainsi que leur représentation.
Partie 4	**Informations utilisateur** Directives pour l'utilisateur, assistant ce dernier lors de la phase de réalisation de l'application.
Partie 5	**Services de communication** Définitions relatives à la communication interne et à l'intercommunication entre les différents partenaires de communication d'une application.

Littérature et références concernant la norme CEI 61131

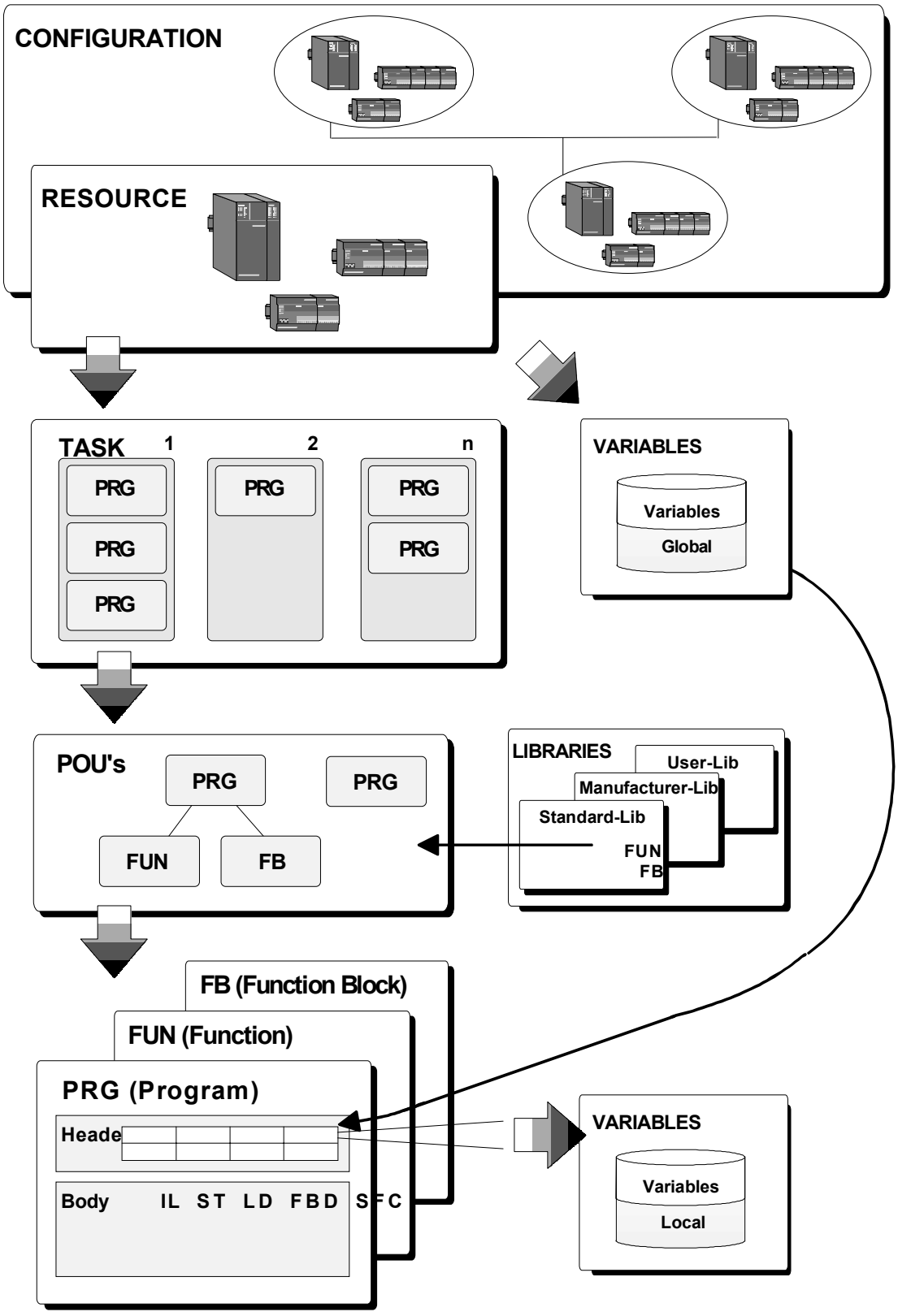
Pour tout approfondissement du sujet, nous vous recommandons les ouvrages suivants :

Titre	Editeur	ISBN
Norme API : CEI 61131	Oldenbourg	3-486-27005-2
Programmation des API avec CEI 61131-3	Springer	3-540-66445-9
Développement de Software pour API selon CEI 61131	Huetig	3-7785-2681-2

En ce qui concerne l'application de la norme CEI 61131, il existe une association d'utilisateurs internationale :

PLCopen Standardization in Industrial Control Programming (Normalisation en matière de programmation de systèmes de commande industriels)	PO Box 2015, NL-5300 CA Zaltbommel Fax: ++31 418 516336 Internet: www.plcopen.org

Vue d‘ensemble des éléments constituant la norme CEI 61131



Configuration, ressource et tâche :

La configuration, la ressource et la tâche décrivent la topologie ou la structure du système de commande. Ces éléments sont également appelés éléments de configuration.

Unités d‘organisation de programme :

Le programme est structuré au moyen de ce que l‘on appelle des POU (unités d‘organisation de programme). Il existe trois types de POU : les programmes, les fonctions et les blocs fonctionnels. Les modules POU disposent d‘une partie comportant les déclarations (appelée en-tête ou « header » - déclaration des données) et d’une partie comportant les instructions (appelée corps ou « body » - programme effectif).

Modules de bibliothèque :

La norme définit des modules normalisés avec interface et fonctionnalité définies. Les modules IEC sont enregistrés dans la bibliothèque standard. En outre, il existe une bibliothèque du fabricant (POU spécifique au fabricant) et la bibliothèque utilisateur (bibliothèques programmées par l’utilisateur).

Variables locales et globales :

La norme distingue les variables locales et les variables globales. Les variables globales sont déclarées au niveau de la ressource et peuvent être mises en œuvre de façon globale dans les POU. Les variables locales sont définies dans une POU (en-tête) et ne peuvent être utilisées que dans cette même POU. L’existence de variables locales permet un encapsulage fiable des données, ce qui garantit une certaine aptitude à la maintenance et à la réutilisation des programmes ou modules de programme.

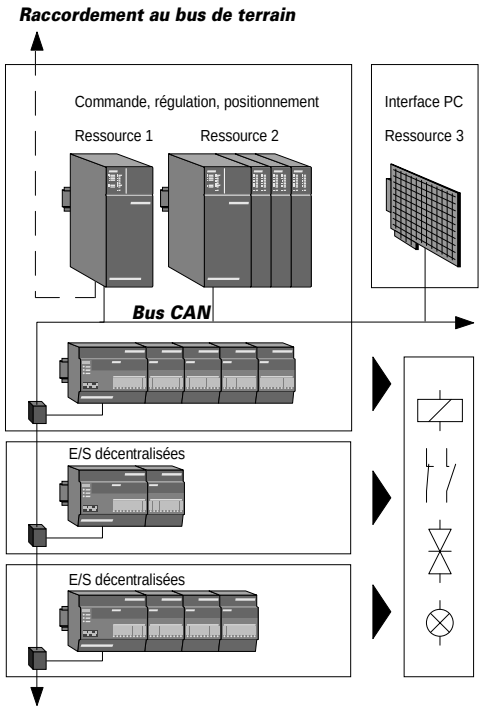
Eléments de configuration

D'un point de vue conceptuel, la norme CEI 61131 part du principe que le système d'automatisation est doté d'une capacité maximale. Ainsi, il est supposé que le système est un système multiprocesseurs pourvu d'une quantité infinie de mémoire, d'un nombre infini de canaux d'entrées/sorties, offrant un nombre infini d'interfaces de communication et reposant sur un système de gestion multitâche.

Les éléments de configuration suivants sont définis par la norme CEI 61131:

Configuration	La configuration est le moyen hiérarchiquement supérieur pour définir la structure d'un système d'automate et ses sous-systèmes.
Ressource	Une configuration peut être constituée de plusieurs ressources (sous-systèmes API), communiquant entre elles sous une forme quelconque.
Tâche	Une ressource peut contenir plusieurs tâches (en anglais « task »). Une tâche commande l'exécution des différents éléments de programme qui lui sont attribués. Chaque tâche peut présenter des caractéristiques d'exécution différentes : il existe des tâches à exécution cyclique, commandée en temps ou commandée par événement.

Exemple d'une configuration avec 3 ressources :



Organisation des programmes

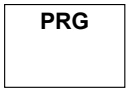
Un programme CEI 61131 est organisé au travers d'unités d'organisation de programme.

Ces unités d'organisation de programme (POU) sont généralement constituées d'une partie déclarative (en-tête) et d'une partie comportant les instructions (corps).

- La partie déclarative contient les données – c.-à-d. toutes les variables, les drapeaux, entrées/sorties, qui sont utilisés dans le corps de programme.
- La partie instructions contient le programme effectif. Celle-ci est programmée dans l'un des langages IEC LT, TS, SR, LOG ou GRAFCET.

Les propriétés des différentes POU sont résumées ci-dessous :

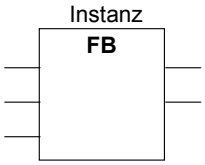
Programme (PRG)



Les programmes sont les véritables programmes principaux, ils sont enregistrés directement dans une tâche. Chaque unité PRG ne peut apparaître qu'une seule fois.

Les programmes peuvent appeler des blocs fonctionnels et des fonctions pour une organisation plus poussée.

Bloc fonctionnel (FB)

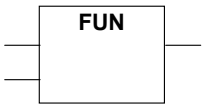


Un bloc fonctionnel est appelé par un programme ou par un autre bloc fonctionnel.

Les blocs fonctionnels peuvent présenter plusieurs entrées / sorties au travers desquelles les valeurs de paramètre sont transférées. Un même bloc fonctionnel peut être utilisé plusieurs fois et indépendamment les uns des autres. Chaque « copie » du bloc fonctionnel possède son propre bloc de données qui est conservé en mémoire, même au-delà de la fin de cycle. Le bloc de données est également appelé instance.

Les blocs fonctionnels sont par exemple mis en œuvre pour des compteurs, bascules ou temporisateurs.

Fonction (FUN)



Une fonction est appelée par un programme, un bloc fonctionnel ou par une autre fonction.

Les fonctions disposent d'une ou de plusieurs entrées, mais d'une seule sortie. Les fonctions peuvent également être utilisées plusieurs fois, néanmoins elles ne disposent pas d'une « mémoire » : chaque appel est traité avec de nouvelles données qui sont perdues lorsque la fonction est quittée.

Les fonctions sont p. ex. mises en œuvre pour les calculs mathématiques (sin, sqrt) ou pour les conversions de type (bcd_to_int, byte_to_int).

Langages de programmation

La partie 3 de la norme spécifie la syntaxe et la sémantique des langages de programmation.

Langages textes	LI	Liste d'instructions	(IL Instruction List)
	TS	Texte structuré	(ST Structured Text)
Langages graphiques	LOG	Logigramme	(FBD Function Block Diagram)
	SR	Schéma à contacts	(LD Ladder Diagram)
	Grafcet (Langage séquentiel)		(SFC Sequential Function Chart)

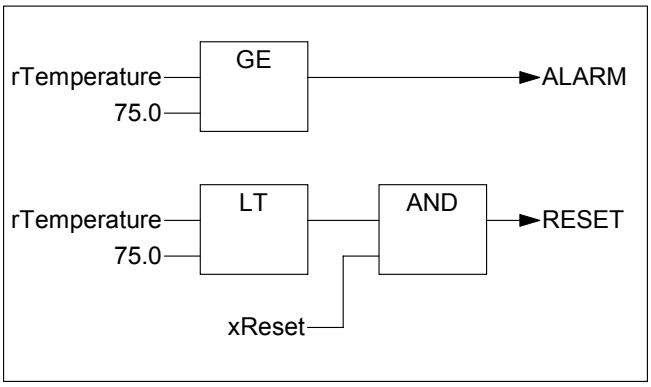
Exemple en LI :

```
        LD      rTemperature    (*over limit*)
        GE      75.0
        JMPC    ALARM

        LD      rTemperature    (*reset*)
        LT      75.0
        AND     xReset
        JMPC    RESET

ALARM:  LD      TRUE            (*set alarm*)
        S       xAlarm
```

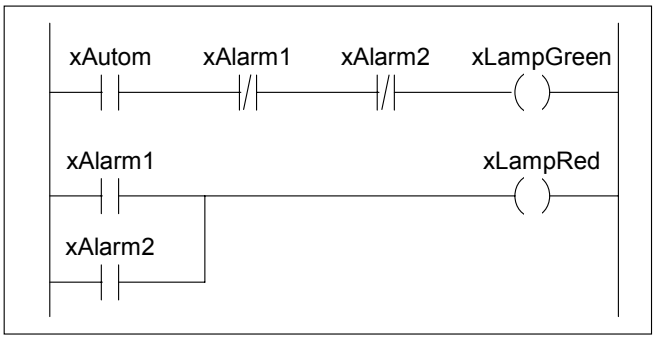
Exemple en TS :

```
(*Set and Reset Marker xAlarm*)

IF   Temperature >= 75.0
THEN
    xAlarm := TRUE;
ELSE
    IF IxReset
    THEN
        xAlarm := FALSE;
    END_IF;
END_IF;
```

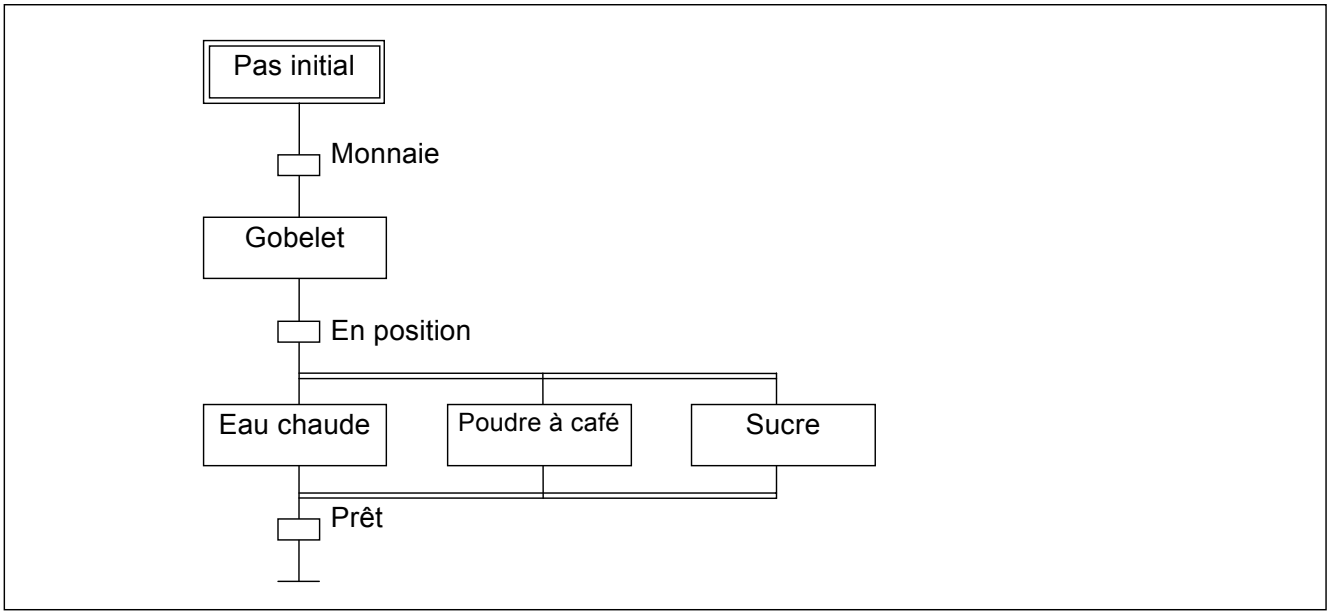
Exemple en LOG :



Exemple en SR :



Exemple en GRAFCET :



Variables et types de données

Représentation des variables

Représentation symbolique

En règle générale, les variables utilisées dans les systèmes IEC ne sont pas désignées par leurs adresses, mais plutôt symboliquement. Ce principe offre quelques avantages importants.

- Réutilisation : des modules ne contenant que des noms de variable symboliques peuvent être très facilement réutilisés dans un nouveau système. Les modifications de câblage nécessaires dans les automates programmables traditionnels sont ainsi évitées.
- Affectation automatique des adresses par le système : dans certains cas rares, il faut effectivement désigner un emplacement mémoire déterminé. Grâce à la représentation symbolique des variables, le rangement des variables en mémoire peut ainsi être pris en charge par le système.

Dans la représentation symbolique, une variable est définie par les éléments suivants :

Class	Type de variable	Cet élément indique s'il s'agit d'une variable globale ou locale, de variables utilisées en tant que paramètres de transfert POU ou de variables rémanentes.
Identifier	Nom de variable	Nom de variable utilisé dans le code programme.
Type	Type de données	Désignation du format des variables : variables binaires (BOOL, BYTE, WORD), variables numériques (INT, REAL) ou texte (STRING).
Initial	Valeur initiale	La variable reçoit cette valeur initiale au démarrage du système de commande.

Variables à représentation directe

Dans le code programme, les variables peuvent également être appelées directement par leurs adresses. Ceci est possible pour certaines zones mémoire globales de la CPU (zone des drapeaux système et drapeaux utilisateur), mais aussi pour les canaux d'entrées/sorties des modules matériels. Pour des raisons de maintenance et de réutilisation des programmes, l'utilisation des variables à représentation directe n'est toutefois pas recommandée.

Les variables à représentation directe sont désignées par leur emplacement physique (entrée/sortie) ou logique (drapeau) et sont munies du préfixe « % ».

Exemples de variables à représentation directe :

% MX 1.127.32	Drapeau binaire 32 dans mot double 127, zone des drapeaux utilisat.
% I X 0.0.1.9	Bit d'entrée 9 du module 1
% Q B 0.0.8.0	Octet de sortie 0 (bits 0 à 7) du module 8

Note : Vous trouverez d'autres informations et exemples concernant l'adressage de variables au chapitre « Comportement du système et zones d'adressage »

Types de variable

Chaque variable doit être attribuée à un type de variable (classe). Il n'est pas possible d'utiliser tous les types de variable dans chaque type de POU. La table ci-dessous donne des éclaircissements quant à l'utilisation et la signification des types de variable selon CEI 61131-3

Type :	Utilisation dans :				Description :
	FUN	FB	PRG	GVL	
VAR_GLOBAL				x	Définition globale d'une variable dans la liste des variables globales (GVL).
VAR_GLOBAL_ RETAIN				x	Version démarrage à chaud de VAR_GLOBAL
VAR_GLOBAL_ CONST				x	Variable globale pouvant être uniquement lue dans le programme (en revanche, l'écriture est possible par le biais de l'outil de programmation = Entry-Data-Monitor)
VAR_GLOBAL_ CONST_RETAIN				x	Version démarrage à chaud de VAR_GLOBAL_CONST

Type :	Utilisation dans :				Description :
	FUN	FB	PRG	GVL	
VAR_EXTERNAL		x	x		Déclaration « EXTERNAL » d'une variable VAR_GLOBAL (une variable globale {GVL, IO-Punkt} utilisée dans une POU est déclarée « EXTERNAL » dans l'en-tête de la POU.)
VAR_EXTERNAL_RETAIN		x	x		Déclaration « EXTERNAL » d'une variable VAR_GLOBAL_RETAIN
VAR_EXTERNAL_CONST		x	x		Déclaration « EXTERNAL » d'une variable VAR_GLOBAL_CONST
VAR_EXTERNAL_CONST_RETAIN		x	x		Déclaration « EXTERNAL » d'une variable VAR_GLOBAL_CONST_RETAIN
VAR	x	x	x		Définition locale d'une variable (dans un en-tête POU)
VAR_RETAIN	x	x	x		Version démarrage à chaud de VAR
VAR_CONST	x	x	x		Variable locale pouvant être uniquement lue dans le programme (en revanche, l'écriture est possible par le biais de l'outil de programmation = Entry-Data-Monitor)
VAR_CONST_RETAIN	x	x	x		Version démarrage à chaud de VAR_CONST
VAR_INPUT	x	x			Variable d'entrée transférée à une POU
VAR_OUTPUT		x			Variable de sortie provenant d'une POU
VAR_OUTPUT_RETAIN		x			Version démarrage à chaud de VAR_OUTPUT
VAR_IN_OUT		x			Paramètre modifiable transféré à une POU (simultanément entrée et sortie)

Types de données

Types de données élémentaires et génériques

La norme met à disposition un certain nombre de types de données élémentaires. Ces types de données peuvent également être regroupés et utilisés dans une application en tant que types de données généraux (ou types de données génériques).

Les types de données suivants sont disponibles :

Types de données génériques :	Types de données élémentaires :	Désignation :	Réso-lution :	Plage :
ANY_BIT	Boolean	**BOOL**	1 bit	FALSE / TRUE
	Byte (octet)	**BYTE**	8 bits	2#00000000 ... 2#11111111
	Word	**WORD**	16 bits	16#00 ... 16#FF
	Double Word	**DWORD**	32 bits	16#0000 ... 16#FFFF
	Long Word	**LWORD**	64 bits	non implémenté
ANY_NUM	Short Integer	**SINT**	8 bits	-128 ... 127
	Integer	**INT**	16 bits	-32768 ... 32767
	Double Integer	**DINT**	32 bits	-2147483648 ... 2147483647
ANY_INT	Long Integer	**LINT**	64 bits	non implémenté
	Unsigned Short Integer	**USINT**	8 bits	0 ... 255
	Unsigned Integer	**UINT**	16 bits	0 ... 65535
	Unsigned Double Integer	**UDINT**	32 bits	0 ... 4294967295
	Unsigned Long Integer	**ULINT**	64 bits	non implémenté
	Real	**REAL**	32 bits	1.2E-38 ... 3.4E38
ANY_REAL	Long Real	**LREAL**	64 bits	non implémenté
	Time (heure)	**TIME**	32 bits (1ms)	T#-596h31m23s648ms ··· T#596h31m23s647ms
ANY_DATE	Date (date)	**DATE**	32 bits (1s)	non implémenté
	Time of Day (heure)	**TIME_OF_ DAY**	32 bits (1s)	non implémenté
	Date and Time	**DATE_ AND_TIME**	32 bits (1s)	DT#1970-01-01-00:00:00 ... DT#2106-02-07-06:28:15
	String (Text)	**STRING[255]**	256 carac.	'Ceci est un exemple de variable texte'

Note :

CAP 1131 supporte tous les types de données jusqu'à 32 bits sans « DATE » et « TIME_OF_DAY ».

Array (tableau) et structure

Sur la base des types de données élémentaires, il est possible de créer certains types de données que l'on appelle types structurés.

- **Array**

  Une variable du type « Array » contient plusieurs éléments d'un même type de données élémentaire auxquels il est possible d'accéder au moyen d'indices. Un tableau peut être unidimensionnel, bidimensionnel ou tridimensionnel.

  Exemple :

Définition de variable : (écriture IEC formelle)	VAR asTexte: **ARRAY [0..9] OF STRING[20]**; arPosition: **ARRAY [1..10, 1..5] OF REAL**; iIndex: **INT**; END_VAR		
Utilisation dans le programme : (en LI)	LD ... LD ST LD ... LD	asTexte[2] 5 iIndex asTexte[iIndex] arPosition[7,2]	(* lecture 2ème élément *) (* lecture indirecte *) (* du 5ème élément *) (* lecture de l'élément *) (* du tableau à 2 dim. *)

- **Structure**

  Une variable de ce type contient plusieurs éléments pouvant être adressés par leur nom. Concernant les éléments, il peut s'agir de types de données élémentaires ou même de tableaux.

  Exemple :

Définition de type: (écriture IEC formelle)	TYPE **ANALYSE**: STRUCT rMesureMax: **REAL**; rMesureMin: **REAL**; rMesureTot: **REAL**; rMoyenne: **REAL**; uiNombreMesures: **UINT**; END_STRUCT; END_TYPE
Définition de variable : (écriture IEC formelle)	VAR Point_mesure1: **ANALYSE**; Point_mesure2: **ANALYSE**; END_VAR
Utilisation dans le programme : (exemple en LI)	LD Point_mesure1.rMesureTot ... ST Point_mesure2.rMoyenne

Note :

Au lieu de l'écriture IEC formelle TYPE ... END_TYPE ou VAR ... END_VAR, il est également possible de présenter la définition des variables de façon plus claire, sous forme tabellaire. A cette fin, CAP 1131 propose des éditeurs de variables/types confortables :

Définition de type : dans CAP 1131, les types de données structurés sont créés dans une unité appelée « DUT_Pool » (DUT= **D**ata **U**nit **T**ype)

Définition de variable : définition globale des variables dans la liste GVL (**G**lobal-**V**ariable-**L**ist)

Note :

Les types de données structurés peuvent également être dotés de valeurs initiales. A cette fin, CAP 1131 met à disposition des boîtes de dialogue spéciales.

Entrées directes de nombres, textes et temps

La syntaxe pour la représentation et l'entrée de données relatives au temps, de valeurs numériques et de textes est clairement définie par la norme. Les types suivants sont distingués :

Valeurs numériques

Valeurs binaires :

Type de données :	Type de représentation :	Exemples :
BOOL	booléen	0, 1, FALSE, TRUE
BYTE, WORD, DWORD	binaire décimal octal hexadécimal	2#00001101 13 8#015 16#0D

Nombres entiers et nombres réels :

Type de données :	Type de représentation :	Exemples :
INT, DINT, ...	décimal	-23; 0; 45
UINT, UDINT, ...	décimal	0; 8; 233
REAL	virgule flottante virgule flottante, exposant	-12.5; 0.025; 3.14159 -1.66E+12; 4.5E-33; 24.0E+17

Note :

La représentation décimale se conforme à la norme IEC 599 « Binary floating point arithmetic for microprocessors » (Arithmétique binaire en virgule flottante pour microprocesseurs).

Textes

Les chaînes de caractères fixes sont des séquences de 0 à x caractères écrites entre guillemets simples ('). Tous les caractères de texte de la table ISO 646 s'utilisent de cette manière.

Règles expliquées au moyen d'exemples :

Chaîne de caractères	Explication
''	Chaîne vide
'Bonjour'	Chaîne de longueur 7 contenant le texte *Bonjour*
'Veuillez entrer une valeur :'	Chaîne de longueur 28 contenant le texte *Veuillez entrer une valeur:*
'$0D$0A'	Chaîne de longueur 2 contenant <CR> et <LF>
'$$'	Chaîne de longueur 1 contenant le texte *$*
'$''	Chaîne de longueur 1 contenant le texte '

Exemple :

```
LD      Textvar1        (*La variable du type String Textvar1*)
EQ      'Bonjour'       (*est comparée avec le texte 'Bonjour'*)
```

Temps

Durée :

L'on entend par durée un intervalle de temps donné, indépendant de l'heure courante. La durée peut également être négative.

Type de données :	Type de représentation :	Exemples :
TIME	en millisecondes	t#100ms (ou T#100ms ou TIME#100ms)
	en secondes	t#10s ; t#-15.5s ; t#63s
	en minutes	t#20m ; t#19.5m
	en heures	t#-5h ; t#4.2h ; t#30h
	en jours	t#3d ; t#-2.5d
	représentation mixte	t#3h10m8.5s

Heure et date:

Contrairement à la durée, ces variables se réfèrent à l'heure et la date courantes.
Il n'y a pas de valeurs négatives !

Type de données :	Type de représentation :	Exemples :
DATE_AND_TIME, ...	Heure et date	dt#1998-06-20-17:05:23 (ou DT#... ou DATE_AND_TIME#...)

Eléments communs

Dans ce paragraphe sont définis les éléments de texte et graphiques qui sont valables pour tous les langages de programmation.

Jeu de caractères

Les caractères utilisés pour la programmation sont ceux définis dans la table des codes de base ISO 646.

Identificateurs

Un identificateur est le nom symbolique d'un des éléments suivants :

- variables
- unités POU (programmes, fonctions et blocs fonctionnels)
- tâches

Les identificateurs se composent d'une suite de caractères alphabétiques, numériques et du caractère de soulignement. Le premier caractère est toujours une lettre.

Règles s'appliquant aux identificateurs

Caractères autorisés	Plage	Remarque
Caractères alphabétiques	A à Z, a à z	Il n'y a pas de distinction entre les lettres majuscules et les lettres minuscules. Par exemple, 'abcd' ou 'ABCD' ou 'aBCd' sont interprétées de la même manière.
Caractères numériques	0 à 9	Un identificateur ne doit pas commencer par un chiffre. Le premier caractère doit être une lettre.
Caractère de soulignement	_	Les caractères de soulignement apparaissant dans les identificateurs sont significatifs. Par exemple, 'A_BCD' n'est pas le même identificateur que 'AB_CD'. La définition de plusieurs caractères de soulignement consécutifs n'est pas autorisée.

Caractères non autorisés	Plage
Espace (Space)	
Crochets et parenthèses	()[]{}
Caractères spéciaux	¦@#%&ɔ*
Signes de ponctuation	,.:;?!+–/<=>≅
Symboles monétaires	$£
Caractères nationaux	¿ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏÐÑÒÓÔÕÖØÙÚÛÜÝßÞàáãâäæçèéêë ìíîïðñòóôõöøùúûüýþÿ

Mots-clés

Les mots-clés sont des combinaisons uniques de caractères et sont utilisés en tant qu'éléments syntaxiques individuels.

Tous les mots-clés sont réservés et ne **peuvent pas** être attribués à des identificateurs. Les mots-clés concernent les domaines suivants :

Caractères définissant une action (qualificatifs)	N,S,R,P,D,DS,SD,SL etc.
Description de configuration	CONFIGURATION, RESOURCE, TASK etc.
Définition de variables	AT, VAR, etc.
Eléments de langages de programmation	LD, ST, CASE, etc.
Noms de types de données	BOOL, INT, REAL etc.
Noms de fonctions	abs, sin, etc.
Noms de blocs fonctionnels	TON,CTU etc.
Enable (validation)	EN,ENO
Etats booléens	FALSE,TRUE
Paramètres de blocs fonctionnels et de fonctions standard	S1, R1 etc.

Commentaires

Les commentaires peuvent être créés par l'utilisateur selon les structures suivantes :

Structure monoligne : (* Texte de commentaire *)

Structure multiligne : (* Un commentaire peut être entré
sur plusieurs lignes *)

Renvois aux normes utilisées

Les normes suivantes contiennent des dispositions qui ont été reprises par la norme CEI 61131 au moyen de renvois. Au moment de la publication de la norme, les éditions suivantes ont été valables.

Norme	Description	Edition
IEC 50	Glossaire international d'électrotechnique	
IEC 559	Arithmétique binaire à virgule flottante pour microprocesseurs	1989
IEC 617-12	Symboles graphiques pour représentations graphiques, partie 12 : éléments logiques binaires	1991
IEC 617-13	Symboles graphiques pour représentations graphiques, partie 13 : éléments analogiques	1978
IEC 848	Préparation de tableaux de fonctions pour systèmes de commande	1988
ISO/AFNOR	Dictionnaire d'informatique, ISBN 2-12-486911-6	1989
ISO/IEC 646	Technologie de l'information - Jeu de caractère ISO codé sur 7 bits pour l'échange de données dans les systèmes de traitements de l'information	1991
ISO 8601	Eléments de données et échanges de formats - Echanges d'informations - Représentation de dates et heures	1988
ISO 7185	Technologie de l'information - Langage de programmation PASCAL	1990
ISO 7498	Systèmes de traitement de l'information - Interconnexion de systèmes ouverts - Modèle de référence de base	1984