SIEMENS

WinCC

Configuration Manual

Manuel

Tome 1

C79000-G8277-C139-01

Edition Février 1999

C79000-G8277-C139-01
Printed in the Federal Republic of Germany

Siemens Aktiengesellschaft

Table des matières

1 Manuel de configuration 1-1
1.1 Manuel de configuration - Structure et utilisation 1-2

2 WinCC - Remarques générales 2-1
2.1 WinCC - le concept 2-2
2.1.1 Les interfaces de WinCC 2-3
2.2 WinCC - Termes et leur explication 2-5

3 Configuration - Thèmes généraux 3-1
3.1 Avant de commencer un projet 3-2

3.2 Détail des conventions 3-3
3.2.1 Convention : Noms de projet WinCC 3-4
3.2.2 Convention : Noms de variable 3-5
3.2.3 Convention : Noms de vue 3-7
3.2.4 Convention : Scripts et actions 3-9
3.2.5 Convention : L'interface utilisateur 3-10
3.2.6 Convention : Concept de conduite 3-15
3.2.7 Convention : Définition des couleurs 3-18
3.2.8 Convention : Les cycles de rafraîchissement 3-19
3.2.9 Convention : Les droits d'accès 3-20
3.2.10 Convention : Alarmes 3-21
3.2.11 Convention : Réalisation 3-22

3.3 Particularités de la configuration avec WinCC 3-23

3.3.1 Cycle de rafraîchissement - Où et comment s'effectue le paramétrage 3-24
3.3.1.1 Rafraîchissement dans une vue 3-24
3.3.1.2 Types de cycle de rafraîchissement 3-27
3.3.1.3 Signification des cycles de rafraîchissement 3-28
3.3.1.4 Remarques sur l'utilisation des cycles de rafraîchissement 3-30
3.3.1.5 Exécution de scripts en arrière-plan (Global Script) 3-34

3.3.2 Dynamisation WinCC 3-38
3.3.2.1 Dynamisation de propriétés 3-38
3.3.2.2 Dynamisation d'événements 3-39
3.3.2.3 Modes de dynamisation des objets 3-39

3.3.3 Environnement système WinCC 3-42
3.3.3.1 Structure de répertoires du système WinCC 3-42

3.3.4 Environnement projet WinCC 3-46
3.3.4.1 Projet WinCC - Structure de répertoires 3-46

3.3.5 Lancement automatique de projet par WinCC 3-49
3.3.6 Arrêt ordonné de WinCC 3-52
3.3.7 Sauvegarde des données 3-53
3.3.8 Copie d'un projet WinCC sauvegardé sur une machine cible 3-55

3.3.9 Réutilisation - Reprise d'éléments de projet dans un nouveau projet ou dans un projet existant 3-57
3.3.9.1 Réutilisation de vues 3-58
3.3.9.2 Réutilisation de symboles et de bitmaps 3-60
3.3.9.3 Réutilisation d'une bibliothèque de projets (contenant des symboles et des objets utilisateur préconfigurés) 3-61
3.3.9.4 Réutilisation d'actions 3-64
3.3.9.5 Réutilisation de variables 3-65
3.3.9.6 Réutilisation de textes multilingues (dans vues, alarmes) 3-72
3.3.9.7 Réutilisation d'alarmes 3-74
3.3.9.8 Réutilisation de valeurs de mesure 3-77
3.3.9.9 Réutilisation de modèles d'impression 3-77
3.3.9.10 Réutilisation d'actions globales 3-77
3.3.9.11 Réutilisation de fonctions de projet 3-78
3.3.9.12 Utilisation de fonctions standard 3-78
3.3.9.13 Réutilisation de la gestion des utilisateurs 3-78

3.3.10 Commande sans souris 3-79
3.3.10.1 Commande par clavier 3-79
3.3.10.2 Déplacement dans les objets de commande (champs de saisie et champ de commande) 3-85
3.3.10.3 Touches de fonction d'Alarm Logging comme touches de barre d'outils 3-87
3.3.10.4 Boutons de barre d'outils Alarm Logging spécifiques au process 3-90
3.3.10.5 Boutons de fonction Tag Logging comme boutons de barre d'outils 3-90
3.3.10.6 Déclenchement de travaux d'impression 3-92
3.3.10.7 Connexion au système ou déconnexion 3-94

3.3.11 Technique des composants de vue 3-95
3.3.11.1 Boîte de process comme composant de vue 3-96
3.3.11.2 Composant de vue avec adressage indirect 3-99
3.3.11.3 Objets utilisateur 3-100
3.3.11.4 Instanciation dynamique 3-101
3.3.11.5 Vues prototypes 3-102
3.3.11.6 Objets OCX 3-107

3.3.12 Configuration en ligne (runtime) - Remarques, restrictions 3-108

4 Cours de programmation en C - WinCC 4-1
4.1 Environnement de développement de scripts dans WinCC 4-3
4.1.1 Fonctions et actions dans WinCC 4-4
4.1.2 Les éditeurs de fonctions et d'actions 4-8
4.1.3 Création de fonctions et d'actions 4-10
4.1.4 Test des fonctions et actions 4-12
4.1.5 Importation et exportation de fonctions et actions 4-14

4.2 Variables en C 4-15
4.2.1 Exemple 1 - Utilisation de types de variable 4-18
4.2.2 Exemple 2 - Variable C en liaison avec des variables de WinCC 4-20
4.2.3 Exemple 3 - Utilisation de variables 4-22

4.3 Opérateurs et fonctions mathématiques en C 4-23
4.3.1 Algèbre booléenne 4-25
4.3.1.1 Fonctions logiques de base 4-25

4.3.2 Exemple 1 - Utilisation des opérateurs des opérations arithmétiques de base 4-27
4.3.3 Exemple 2 - Fonctions mathématiques 4-28
4.3.4 Exemple 3 - Opérateur pour opérations binaires 4-30

4.4 Pointeur en C 4-32
4.4.1 Exemple 1 - Pointeurs 4-34
4.4.2 Exemple 2 - Pointeurs en liaison avec des variables de WinCCpointeur 4-36
4.4.3 Exemple 3 - Pointeurs en liaison avec l'exécution de chaînes de caractères 4-38

4.5 Boucles et instructions conditionnelles en C 4-39
4.5.1 Exemple 1 - Boucle do - while 4-42
4.5.2 Exemple 2 - Boucle do - while 4-44
4.5.3 Exemple 3 - Boucle for 4-46
4.5.4 Exemple 4- Instructions conditionnelles avec if 4-48
4.5.5 Exemple 5- Instruction conditionnelle avec switch et case 4-50
4.5.6 Exemple 6- Utilisation de variables statiques avec une instruction conditionnelle et valeur retournée 4-52
4.5.7 Exemple 7- Utilisation de variables statiques avec une instruction conditionnelle et valeur retournée 4-53

4.6 Opérations sur fichiers en C 4-54
4.6.1 Exemple 1 - Ouvrir, écrire et fermer un fichier 4-56
4.6.2 Exemple 2 - Ouverture, extension et fermeture d'un fichier 4-58
4.6.3 Exemple 3 - Ouvrir, lire et fermer un fichier 4-60
4.6.4 Exemple 4 - Supprimer un fichier 4-62

4.7 Structures en C 4-63
4.7.1 Exemple 1 - Structures en C 4-65
4.7.2 Exemple 2 - Structures en C en liaison avec WinCC 4-67

4.8 Global Scripts 4-69
4.8.1 Exemple 1 - Utilisation d'une fonction de projet 4-70
4.8.2 Exemple 2 - Utilisation de fonctions de projet, autres exemples 4-72

4.9 Exemple de projet 4-73
4.10 Affichage du code source avec le bouton droit de la souris 4-74

5 Annexe 5-1
5.1.1 Entrée/Sortie normalisées dans champ E/S 5-2
5.1.2 Actions spécifiques à des objets à la sélection de vue 5-3
5.1.3 WinCC-Scope 5-4
5.1.4 Accès à la base de données 5-5
5.1.4.1 Accès à la base de données avec Excel/MSQuery 5-5
5.1.4.2 Accès à la base de données avec Access 5-8
5.1.4.3 Accès à la base de données avec ISQL 5-9
5.1.4.4 Accès à la base de données avec WinCC-Scope 5-10
5.1.4.5 Exportation à partir de la base de données avec des actions C 5-10
5.1.4.6 Sélection de tables dans la base de données 5-12
5.1.5 Couplage série 5-13
5.1.6 Table des couleurs 5-14

5.2 Documentation du système d'alarmes S5 5-15

5.2.1	Liste des blocs logiciels	5-16
5.2.2	Conditions à remplir par le matériel	5-17
5.2.3	Intégration du système d'alarmes S5 dans le programme utilisateur SIMATIC-S5	5-18
5.2.4	Description générale du système d'alarmes S5s	5-20
5.2.4.1	Structure du bloc de données d'offset	5-20
5.2.4.2	Numéro d'alarme de base	5-21
5.2.4.3	Numéro d'offset / Etats des signaux des alarmes	5-22
5.2.4.4	Bloc d'états de signaux	5-23
5.2.4.5	Adresse du dernier bloc d'états de signaux	5-24
5.2.4.6	Etats de signaux	5-25
5.2.4.7	Etats de repos	5-25
5.2.4.8	Bits d'acquittement	5-26
5.2.4.9	Mémentos de fronts de signaux	5-26
5.2.4.10	Structure du bloc de données de paramètres	5-26
5.2.4.11	Structure d'un bloc d'alarme	5-27
5.2.4.12	Numéro d'alarme	5-28
5.2.4.13	Etat d'alarme	5-28
5.2.4.14	Horodatage	5-28
5.2.4.15	Variables de process	5-29
5.2.4.16	Numéro d'ordre de fabrication / désignation de charge	5-29
5.2.4.17	Réserve	5-29
5.2.4.18	Formation d'un bloc d'alarme	5-29
5.2.4.19	Tampon cyclique interne (FIFO)	5-29
5.2.4.20	Interface de transfert de données au système WinCC	5-30
5.2.5	Description d'interfaces	5-31
5.2.5.1	Bloc de données système 80	5-31
5.2.5.2	Bloc de données d'offset	5-31
5.2.5.3	Bloc de données de paramètres	5-31
5.2.5.4	Interface de transfert	5-31
5.2.6	Paramétrage du système d'alarmes S5 / DB Système 80	5-32
5.2.7	Exemples de configuration du système d'alarmes S5	5-37
5.2.7.1	Paramétrages DB 80	5-37
5.2.7.2	Création des blocs de données	5-38
5.2.7.3	Initialisation des blocs de données d'offset	5-38
5.2.8	Documentation des blocs de commande SIMATIC S5	5-42
5.2.8.1	Liste des blocs logiciels	5-42
5.2.8.2	Conditions à remplir par le matériel	5-43
5.2.8.3	Paramètres d'appel de FB 87 : EXECUTE	5-43
5.2.9	Description d'interfaces	5-44
5.2.9.1	Exemples de configuration pour les blocs de commandes S5	5-45
5.2.10	Fonctions et propriétés de la synchronisation de l'heure S5	5-46
5.2.10.1	Liste des blocs logiciels	5-46
5.2.10.2	Conditions à remplir par le matériel	5-46
5.2.11	Paramètres d'appel de FB 86 : MELD:UHR	5-47
5.2.12	Formats de données pour l'heure et la date	5-49
5.2.12.1	Zone des données d'heure CPU 944, CPU 945	5-50
5.2.12.2	Zone de données d'heure CPU 928B, CPU 948	5-51
5.2.12.3	Zone des données d'heure CPU 946, CPU 947	5-52
5.2.12.4	Format de données d'heure pour les blocs d'alarmes	5-53
5.2.13	Description d'interfaces	5-54
5.2.14	Interaction avec le système d'alarmes WinCC	5-55

5.3	DLL de normalisation d'interface vers AlarmLogging et TagLogging	5-57
5.3.1	Interface commune vers AlarmLogging et TagLogging	5-58
5.3.2	Compléments spécifiques à TagLogging	5-60
5.3.3	Fonctions API d'une DLL de normalisation WinCC	5-61
5.3.3.1	Initialisation de la DLL de normalisation	5-61
5.3.3.2	Interrogation des propriétés d'une DLL de normalisation	5-61
5.3.3.3	Interrogation du nom de la DLL de normalisation	5-64
5.3.4	Fermeture de la DLL de normalisation	5-65
5.3.4.1	Extensions de la configuration	5-65
5.3.4.2	Extension du dialogue de configuration d'alarmes S7PMC	5-65
5.3.4.3	Extension du dialogue de configuration de variables d'archive	5-67
5.3.4.4	Services en ligne	5-69
5.3.4.5	Enregistrer toutes les variables d'archive	5-69
5.3.4.6	Commutation de langue	5-71
5.3.5	Normalisation	5-72
5.3.5.1	Dérivation d'alarmes individuelles	5-72
5.3.5.2	Acquittement, verrouillage/autorisation d'alarmes	5-73
5.3.5.3	Exécution sur changement d'état	5-74
5.3.5.4	Mise à jour des alarmes de la DLL de normalisation S7PMC	5-75
5.3.5.5	Normalisation de variables d'archive	5-75
5.3.5.6	Dérivation de valeurs de variables d'archive individuelles	5-75
5.3.5.7	Verrouillage / autorisation de variables d'archive	5-77
5.3.5.8	Exécution sur changement d'état	5-77
5.4	Composants de vue avec Visual Basic 5.0	5-79
5.5	Bibliothèque globale	5-81
5.5.1	Shut off devices	5-82
5.5.2	Shut off valves	5-83
5.5.3	Objet utilisateur	5-84
5.5.3.1	Afficheurs	5-84
5.5.3.2	Pupitre de Commande	5-84
5.5.3.3	Bouton-Incr-Decr	5-84
5.5.3.4	Tuyau	5-84
5.5.3.5	Cuve	5-85
5.5.3.6	Bouton de Commutation	5-86
5.5.3.7	Vanne	5-86
5.5.3.8	Instrument de Pointage	5-86
5.5.4	Bouton de Image	5-87
5.5.5	Bouton 3D	5-88
5.5.6	Bouton de Langue	5-89
5.5.7	DIN30600	5-90
5.5.8	Symbole E	5-91
5.5.9	Fenetre	5-92
5.5.10	Conveyor	5-93
5.5.11	ISA Symbols	5-94
5.5.11.1	isa_s55a	5-94
5.5.11.2	isa_s55b	5-94
5.5.11.3	isa_s55c	5-94
5.5.11.4	isa_s55d	5-95
5.5.11.5	isa_y32a	5-95
5.5.11.6	isa_y32b	5-95
5.5.11.7	isa_y32c	5-95
5.5.11.8	isa_y32d	5-96

5.5.11.9	isa_y32e	5-96
5.5.11.10	isa_y32f	5-96
5.5.11.11	isa_y32g	5-96
5.5.11.12	isa_y32h	5-97
5.5.11.13	isa_y32i	5-97
5.5.12	Clavier	5-98
5.5.13	Moteur	5-99
5.5.14	Moteur 3D	5-100
5.5.15	PC / PLC	5-101
5.5.16	Pompe	5-102
5.5.17	Régulateur	5-103
5.5.18	Tuyau	5-104
5.5.19	Mise à l'echelle	5-105
5.5.20	Champ de Texte	5-106
5.5.21	Vanne	5-107
5.5.22	Vanne 3D	5-108
5.5.23	Divers1	5-109
5.5.24	Divers2	5-110

Avant-propos

Objet du manuel

Ce manuel présent les possibilités de configuration de WinCC dans les parties suivantes<sp>:

- une partie générale relative à WinCC et à la configuration
- une introduction au traitement des scripts
- une annexe

La table des matières et l'index vous permettront de trouver rapidement les informations dont vous avez besoin. Celles-ci sont bien entendu également disponibles au format HTML dans la documentation en ligne qui propose en outre des fonctions de recherche étentues.

Les exemples de projet illustrant les possibilités de configuration efficace à l'aide WinCC sont décrits dans le *Manuel de configuration WinCC Tome 2*.

Préalables

Connaissance de WinCC (**Getting Started**) ou expérience pratique dans le domaine de la configuration avec mit WinCC.

Informations complémentaires

Pour toute question technique, veuillez vous adresser à votre agence Siemens.

Vous trouverez les adresses des agences et représentations Siemens p. ex. dans l'annexe Siemens dans le monde entier du manuel "Système d'automatisation S7-300, Configuration d'un S7-300", les catalogues et sur Compuserve (go autforum)...

Vous pouvez également nous appeler sur notre ligne directe<sp>:
+49 (911) 895-7000 (Fax 7001).

Vous trouverez également des informations utiles sur Internet à l'adresse suivante<sp>:
www.aut.siemens.de/coros/html_00/coros.htm.

Information sur les produits SIMATIC

Des informations régulièrement mises à jour sur les produits SIMATIC, vous sont proposées<sp>:

- sur Internet sous http://www.aut.siemens.de/
- par appel de télécopie au n° +49 8765–93 02 77 95 00

Le SIMATIC Customer Support met également à votre disposition des informations d'actualité et des téléchargements qui pourront vous être utiles lors de la mise en oeuvre des produits SIMATIC<sp>:

- sur Internet sous http://www.aut.siemens.de/support/html_00/index.shtml
- via le serveur du SIMATIC Customer Support au numéro +49 (911) 895–7100
- Utilisez pour l'accès au serveur un modem transmettant au maximum à 28,8 kBaud (V.34) à paramétrer comme suit<sp>: 8, N, 1, ANSI, ou utilisez l'accès RNIS (x.75, 64 kBit).

Le SIMATIC Customer Support peut être joint par téléphone au n° +49 (911) 895–7000 et par télécopie au n° +49 (911) 895–7002. Vous pourrez également lui faire parvenir vos demandes par courrier électronique via Internet ou par message adressé au serveur mentionné ci-desssus.

1 Manuel de configuration

Ce manuel de configuration, qui fait partie intégrante de la documentation WinCC, est consacré essentiellement à l'utilisation pratique de WinCC dans les projets.

Introduction

Cette introduction apporte au lecteur des informations générales sur la marche à suivre pour réaliser des projets IHM (Interface Homme Machine). Il y a quelques années encore, le système IHM était appelé système de contrôle-commande.

Ces dernières années, les exigences imposées aux systèmes de contrôle de process, ainsi qu'à l'archivage et au traitement subséquent des données de process, sont devenues très sévères. De nouveaux systèmes IHM ont été développés pour répondre à ces contraintes.

WinCC est l'un de ces nouveaux systèmes. WinCC est certainement inégalé en ce qui concerne les fonctionnalités, l'ouverture et l'actualité (l'état de l'art des applications logicielles).

Les systèmes IHM de l'ancienne génération ne proposaient souvent qu'une seule solution aux problèmes à résoudre. WinCC en offre presque toujours plusieurs. Le présent manuel de configuration a été rédigé pour vous permettre de toujours utiliser l'approche optimale quant aux performances et à l'effort de configuration impliqué.

Cette partie de la documentation est destinée à vous proposer diverses solutions en vue d'une utilisation efficace de WinCC dans vos projets.

Nous avons réalisé les solutions proposées dans des projets WinCC servant d'exemples. Ces exemples de projets sont fournis sur le CD-ROM WinCC. Vous pouvez utiliser ces solutions directement dans vos projets et faire ainsi l'économie d'un temps précieux.

1.1 Manuel de configuration - Structure et utilisation

Conditions d'utilisation

Le lecteur devra avoir acquis une expérience pratique de WinCC avant de commencer à utiliser le présent manuel de configuration. **Getting Started** est un moyen de familiarisation idéal pour les débutants en WinCC. «Getting Started» explique les principaux thèmes de manière pratique à l'aide de petits exemples. Ce manuel de configuration vient compléter le système d'aide WinCC (Online et Documentation). Les particularités des objets, propriétés et autres thèmes peuvent être consultées dans le système d'aide WinCC lorsqu'elles ne sont pas traitées dans le présent manuel.

Nota:
Le chapitre *3.3 Particularités de la configuration avec WinCC* décrit l'environnement de projet et les types de dynamisation de WinCC.

Contenu et structure

Le manuel est organisé en six parties principales.

- La première partie est un chapitre général qui contient la préface, l'introduction et les remarques générales sur ce manuel.
- La deuxième partie contient des remarques générales sur WinCC
- La troisième partie contient des remarques générales et spécifiques sur le déroulement structuré et efficace de projets IHM.
- La quatrième partie est un cours de WinCC destiné aux débutants. Celui-ci contient les principales règles à observer pour l'emploi du langage de scripts WinCC. Les habitués du langage C pourront y lire les particularités de l'environnement de développement.
- La sixième partie contient l'annexe avec des thèmes de *Solutions* WinCC et *Trucs & Astuces* WinCC qui n'ont pas été pris en compte dans les exemples de projet.

Conventions

Les conventions ci-dessous sont utilisées dans le présent manuel.

Convention	Description
Important	Les passages isolés et importants sont en caractères gras.
Noms	Les noms de boîtes de dialogue, de boutons et de champs sont en italique.
Entrées	Les entrées sont affichées en italique en bleu.
Menu →Commande de menu →etc.	La sélection par des commandes de menu est affichée en italique en bleu. La flèche indique la séquence des commandes.
`Programme`	Les scripts C sont toujours affichés dans cette forme.
`int`	Les mots-clés sont toujours en bleu dans les scripts C.
`"Texte",'z'`	Dans les scripts C les chaînes de caractères et caractères individuels sont représentés en rouge.
`Commentaire`	Le commentaire est en bleu-vert et toujours affiché avec une taille de police plus petite dans les scripts C.
`Système`	Parties de programme générées par le système et ne devant pas être modifiées
	Clic du bouton gauche de la souris
 **D**	Clic du bouton droit de la souris
 **GG**	Double clic du bouton gauche de la souris
Nota :	Toujours sur fond gris

Où trouver les informations

La **table des matières** contient les informations organisées par rubriques.

L'**index** contient les informations classées d'après les mots-clés.

Presque tous les mots (à l'exception des articles et des mots tels que «et», «ou», etc.) sont rangés par ordre alphabétique dans l'onglet *Rechercher* de la documention **en ligne**. **GG** sur le mot recherché permet d'afficher automatiquement tous les chapitres contenant des occurrences de ce mot.

2 WinCC - Remarques générales

Ce chapitre explique en détail la structure, le concept et le mode d'action de WinCC.

2.1 WinCC - le concept

Du point de vue de la configuration, il existe dans WinCC toujours trois types de solution :

- la configuration avec les outils standard de WinCC,
- l'utilisation d'applications Windows existantes à l'aide de composants DDE, OLE, ODBC et ActiveX,
- le développement d'applications propres (en VisualC++ ou Visual Basic) intégrées dans WinCC.

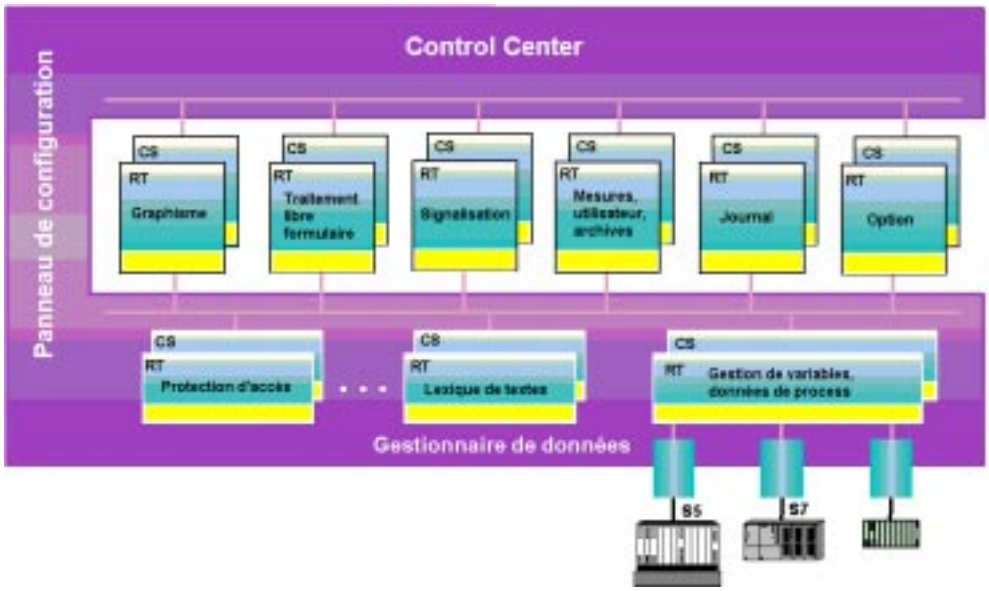
Pour les uns, WinCC représente un système IHM de configuration rapide et économique et, pour les autres, une plate-forme système extensible à l'infini. La modularité et la souplesse de WinCC offrent des possibilités totalement nouvelles pour l'étude et la réalisation de tâches d'automatisation.

Le système d'exploitation: Base de WinCC

WinCC est basé sur le système d'exploitation 32 bits de Microsoft (à l'heure actuelle Windows 95 et Windows NT 4.0). Ces systèmes d'exploitation sont les systèmes d'exploitation standard sur plate-forme PC.

La structure modulaire de WinCC

WinCC propose des composants système pour la visualisation, les alarmes, l'acquisition et l'archivage des données de process ainsi que pour la consignation dans des journaux et l'intégration coordonnée de routines utilisateur librement formulées.
L'utilisateur a également la possibilité d'intégrer ses propres composants.

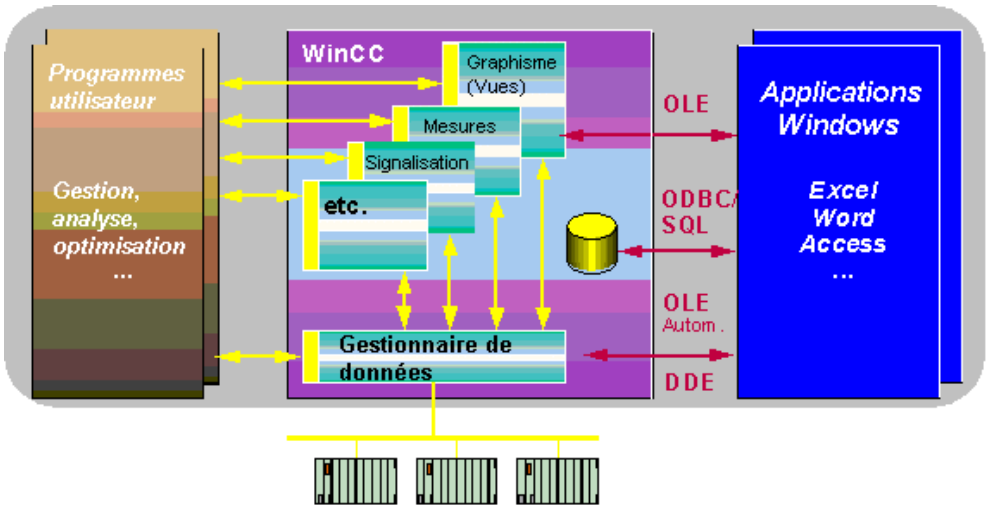


2.1.1 Les interfaces de WinCC

L'ouverture de WinCC

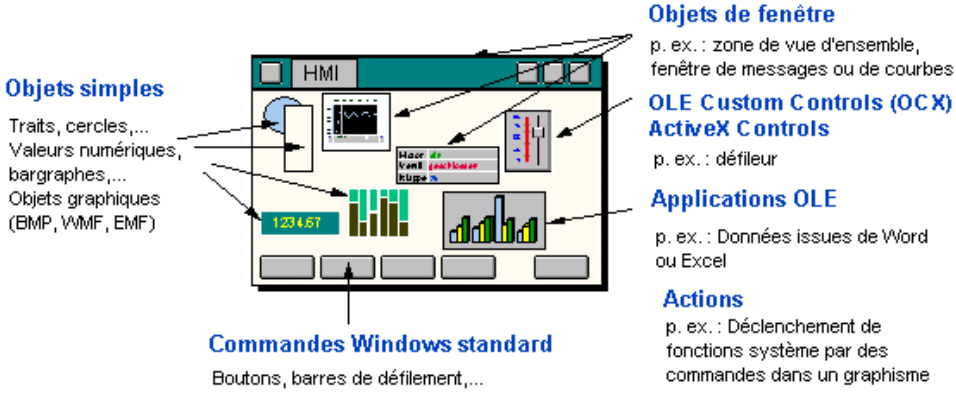
WinCC est totalement ouvert à toute extension par l'utilisateur. Cette ouverture absolue est permise par la structure modulaire de WinCC et par son interface de programmation puissante .

La vue ci-dessous montre les possibilités d'intégrer les diverses applications.



Intégration d'applications tierces dans WinCC

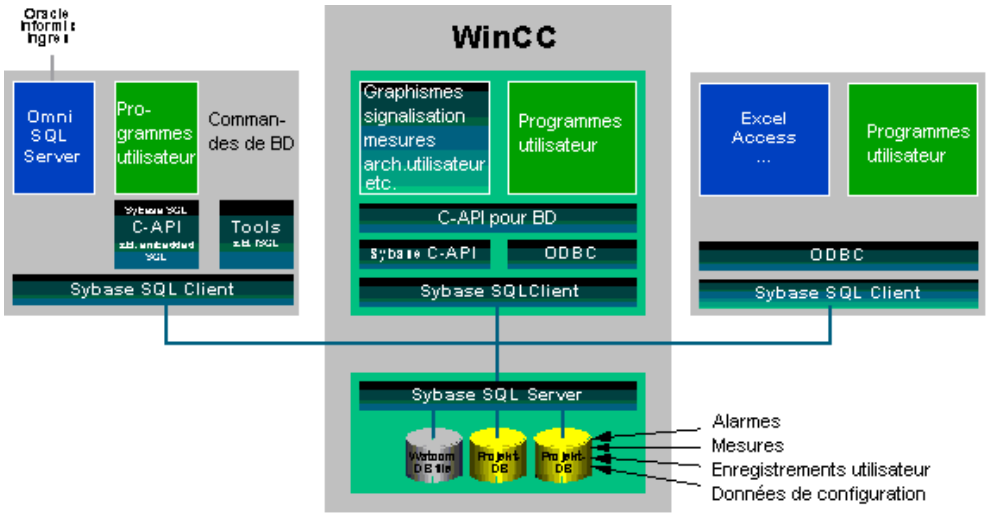
WinCC offre la possibilité d'intégrer, de **manière homogène, des applications et modules applicatifs de développeurs tiers à l'interface utilisateur.**
Comme le montre la figure ci-dessous, WinCC permet d'intégrer à l'application WinCC aussi bien des fenêtres d'application OLE que des contrôles personnalisés OLE (objets OCX 32 bits) et des contrôles ActiveX comme s'il s'agissait d'objets propres à WinCC.



Gestion de données dans WinCC

Dans la figure ci-dessous, WinCC est représenté par toute la partie médiane. Le graphique montre que la **base de données standard Sybase SQL Anywhere** est subordonnée à WinCC. Elle est utilisée pour mémoriser (avec sécurité transactionnelle) toutes les données de configuration de type «listes», telles que listes de variables et listes d'alarmes, mais aussi les données courantes de process tels que les alarmes, valeurs de mesure et enregistrements utilisateur. Cette base de données assume une fonction de serveur. WinCC peut accéder comme client à la base de données via les composants ODBC mais aussi par l'interface de programmation (C-API) ouverte.

Le même droit d'accès est bien entendu accordé à d'autres programmes. Pour cette raison, un **tableur Windows** ou une **base de données Windows** peuvent accéder directement aux données contenues dans la base de données WinCC, que l'application tourne sur le même ordinateur ou sur une station connectée en réseau. Un langage d'interrogation de base de données SQL et les outils de connectivité correspondants (p. ex. pilotes ODBC) permettent également à d'autres clients (p. ex. **base de données UNIX** tels que Oracle, Informix, Ingres) d'accéder aux données de la base WinCC. L'inverse est bien sûr également vrai. Au total, rien ne s'oppose à **l'intégration de WinCC à un concept de niveau usine ou de niveau entreprise**.



2.2 WinCC - Termes et leur explication

Ce chapitre contient une liste alphabétique de tous les termes relatifs à WinCC. Un grand nombre des notions expliquées ici vous sont déjà probablement connues :

IHM	Interface homme-machine
API	Automate programmable industriel
CS	Système de configuration
RT	Runtime

3 Configuration - Thèmes généraux

Le présent chapitre contient de nombreuses remarques et suggestions pour le déroulement de projets avec WinCC. Un certain nombre de ces remarques ne sont pas spécifiques à WinCC.
Dans le cas idéal, ces conventions (règles de configuration) posséderont la qualité d'un guide de configuration et de conception des projets runtime.

3.1 Avant de commencer un projet

Avant de commencer à configurer, vous devez définir certaines conventions et faire un effort de structuration. Ceci permet :

- de simplifier la configuration,
- d'augmenter la clarté du projet,
- de simplifier le travail en équipe,
- d'améliorer la stabilité et les performances et
- de simplifier la maintenance des projets.

La définition de directives de structuration est une condition indispensable à l'élaboration et à l'amélioration d'un standard d'entreprise.

Ces conventions peuvent être classées dans deux catégories :

Conventions pour la configuration

Avant de procéder à la configuration, il conviendra d'établir les conventions suivantes :

- Définir le nom du projet WinCC
- Définir les noms des variables
- Définir les noms des vues WinCC
- Définir les règles de création des scripts et actions
- Définir les règles de configuration (standards d'entreprise, fonction bibliothèque, travail en équipe
- Conventions concernant la documentation du projet

Conventions pour le projet runtime

Il s'agit de conventions pour le projet runtime (résultats de la configuration). Ces conventions dépendent dans une large mesure du domaine d'application (industrie automobile, chimie, machine-outil, etc.). Il conviendra d'établir les conventions suivantes :

- Définir l'interface utilisateur (subdivision de l'écran, type et taille de police, langue du runtime, représentation des objets).
- Définir le concept de conduite (hiérarchie des vues, philosophie de conduite, droits d'accès, boutons autorisés)
- Définir les couleurs des alarmes, valeurs seuils, états, polices, etc.
- Définir les moyens de communication (type de couplage, type et cycle de rafraîchissement)
- Définir les quantités (nombre d'alarmes, valeurs d'archive, courbes, clients, etc.
- Définir les procédures d'alarmes et d'archivage

3.2 Détail des conventions

Nous adoptons dans cette partie du manuel des conventions que nous utiliserons dans nos exemples de projet. Ces conventions sont destinées à vous servir de modèles pour créer vos propres projets.

Nota:
Dans les scripts de nos exemples de projet, les noms des projets, des vues, des variables et les commentaires sont en anglais.

Valeurs par défaut des outils de configuration

Dans la plupart des éditeurs de WinCC, certaines propriétés peuvent être paramétrées par défaut. WinCC supporte ainsi votre propre style de configuration et peut être configuré de manière optimale pour certaines applications.

Nota:
Les possibilités de paramétrage via *Graphics Designer →Outils → Paramètres...* sont indiquées à titre d'exemple. Ce sujet est traité de manière exhaustive dans l'aide de *Graphics Designer*.

3.2.1 Convention : Noms de projet WinCC

Généralités

Le nom du projet est également proposé par défaut pour le répertoire dans lequel sont rangées toutes les données d'un projet WinCC. Le nom de répertoire peut être modifié au moment de la création du projet ou ultérieurement (dans l'explorateur de Windows).

Paramètres / Seuils

Tous les caractères, à l'exception de (p.ex. \ ? ' . ; : /) sont autorisés. Sont également autorisées les valeurs de numérique 0-9.

Convention

Dans les exemples de projets décrits dans le tome 2 du manuel de configuration, les conventions suivantes s'appliquent au nom de projet :

a...a_nn

Signification des lettres :

a Désignation du type (a-z, A-Z, pas de caractères spéciaux).
_n Numéro permettant de distinguer plusieurs projets d'un type (nombres 0-9), intervalle 00-99.

Exemple : cours_00.mcp, ou pictu_01.mcp

Remarque sur l'utilisation généralisée

Le nom de projet WinCC peut par exemple être également utilisé pour distinguer des sous-ensembles d'un process.

Nota:
Les noms de projet WinCC peuvent être imprimés pour la rétrodocumentation. Ceci facilite la mise en correspondance et la recherche d'informations.

3.2.2 Convention : Noms de variable

Généralités

Les noms des variables à attribuer ne sont plus limités à 8 caractères. Les noms de variable doivent cependant ne pas être trop longs. Le respect de règles fixes pour l'attribution de noms de variable comporte des avantages très importants pour la configuration.
Lors de la création de projets WinCC, la construction de la gestion des variables est une des tâches clés conditionnant une configuration rapide et efficace et un traitement performant en runtime (dans des scripts).

Un certain nombre de particularités de la structure de la gestion des variables dans WinCC doivent être prises en compte avant de définir les noms de variable. La création de groupes de variables n'a d'effet que sur la visualisation des variables dans la Gestion des variables pendant la configuration. Les noms de groupe n'ont aucune influence sur l'unicité des noms de variable. Les noms de variables doivent être uniques dans un projet WinCC. L'unicité est vérifiée par le système.

WinCC apporte diverses aides pour la sélection des variables, p. ex. pour le tri en colonnes (noms, date, etc.) ou pour l'utilisation de filtres. Il peut cependant être opportun que les noms de variable contiennent des informations complémentaires.

Convention

Les règles suivantes sont applicables aux noms de variable dans les exemples de projet:

xxxy_z...z_a...a_nn

Signification des lettres :

x Type
- BIN Variable binaire
- U08 Valeur 8 bits non signée *(unsigned)*
- S08 Valeur 8 bits signée *(signed)*
- U16 Valeur 16 bits non signée
- S16 Valeur 16 bits signée
- U32 Valeur 32 bits non signée
- S32 Valeur 32 bits signée
- G32 Nombre à virgule flottante 32 bits IEEE 754
- G64 Nombre à virgule flottante 64 bits IEEE 754
- T08 Variable de texte jeu de caractères 8 bits
- T16 Variable de texte jeu de caractères 16 bits
- ROH Type de données brutes.
- TER Référence texte
- STU Types de structures

y Provenance
- r variable de lecture uniquement issue de l'API (read)
- w variable d'écriture et de lecture issues de l'API (write)
- i variable interne de WinCC, sans liaison à l'API
- x variable à adressage indirect (variable de texte dont le contenu est un nom de variable)

_z groupe (correspond à des sous-ensembles ou bâtiments)
_Lack ... p. ex. nom du sous-semble

_a nom de variable (p. ex. nom d'un point de mesure)
_EU0815V10 ... p. ex. désignation du point de mesure

_n Numéro de l'instance (nombres 0-9) intervalle 00-99.

Paramètres / Seuils

L'attribution des noms de variable est soumise aux restrictions suivantes:

- Le caractère spécial @ devrait être utilisé uniquement pour les variables système WinCC, mais il est également utilisable d'une manière générale.
- L'utilisation des caractères spéciaux ' et % est interdite.
- Le caractère spécial " et les deux caractères successifs // ne doivent pas être employé car ils possèdent une signification particulière dans les scripts C (ils introduisent et terminent une chaîne de caractères ou introduisent un commentaire).
- Pas de blancs
- Le système ne discrimine pas les minuscules des majuscules dans les noms de variable.

Remarque sur l'utilisation généralisée

Les noms attribués aux variables dans les exemples ne sont que des propositions.
Pour l'emploi des variables dans des scripts et dans Excel, il peut être opportun de limiter à une longueur fixe les différents éléments du nom de variable (utiliser le cas échéant des caractères de remplissage 0 ou x).
Excel permet p. ex. de créer et de maintenir avec facilité et efficacité de grandes quantités de variables. Le respect d'une structure fixe dans la constitution du nom de variable facilite la création des listes de variables dans Excel. Ces listes de variables créées dans Excel peuvent être ensuite importées avec le programme *\SmartTools\CC_VariablenImportExport\Var_exim.exe* se trouvant sur le CD WinCC dans le projet WinCC courant.

3.2.3 Convention : Noms de vue

Généralités

Lorsque des vues doivent être appelées dans des scripts ou dans des programmes externes, il peut s'avérer très utile de respecter une structure fixe pour l'attribution des noms de vue. Il est bon de bien réfléchir à la longueur des noms de vue. Les noms trop longs (noms de fichier) ont plutôt tendance à affecter la clarté (sélection dans des zones de liste, appel dans des scripts, etc.). Dans la pratique, une longueur maximale de 40 caractères s'est avérée bien adaptée.

Paramètres / Seuils

L'attribution des noms de vue est soumise aux restrictions suivantes:

- Longueur maximale de 255 caractères.
- Caractères quelconques sauf les caractères spéciaux (p. ex. / " \ : ? < >).:
- Le système ne discrimine pas les majuscules des minuscules.

Convention

Les règles suivantes sont applicables aux noms de vue dans les projets de ce manuel:

aaaaa_k_x...x_nn

Signification des lettres :

a Lettre code (a-z, A-Z, pas de caractères spéciaux) pour les groupes de vues.
cours... p. ex. nom des vues dans cours de C

_k Type de vue, code du type de vue 0 à 99
- _0 Vue de démarrage
- _1 Vue synoptique
- _2 Vue de touches
- _3 Vue de process
- _4 Vue de détail
- _5 Vue d'alarmes
- _6 Vue de courbes
- _7 ...
- _8 ...
- _9 Vues de diagnostic (uniquement pour tests et mise en service)

_x Nom décrivant la fonction de la vue (a-z, A-Z, pas de caractères spéciaux)
30 caractères au maximum.
_chapter ... p. ex. nom du chapitre dans le cours de programmation en C

_n Numéro du type (nombres 0 --à 9) intervalle 0 à 99.

Remarque sur l'utilisation généralisée

Les noms attribués aux vues dans les exemples ne sont que des propositions. Le respect de la convention que nous avons adoptée est cependant la condition de l'utilisation d'une partie des scripts fournies avec le produit.

3.2.4 Convention : Scripts et actions

Généralités

L'utilisateur peut créer ses propres scripts et actions dans les projets WinCC. Choisissez des **noms significatifs**. Ceci s'avérera très utile lorsque vous utiliserez vos scripts.

Pour la configuration dans Global Script (éditeur), l'utilisation d'un type de police proportionnel ne favorise par la perception visuelle. Pour une meilleure lisibilité, il est recommandé d'utiliser une police ayant une largeur de caractère constante (p. ex. Courier).

Les scripts doivent toujours comporter des commentaires suffisants. Le temps passé à créer des commentaires est sans rapport avec celui que l'on perd en essayant de comprendre ultérieurement un programme mal commenté. Bien que suffisamment connu, ceci est souvent ignoré.

Convention

Les règles suivantes sont applicables aux scripts utilisés dans les projets de ce manuel:
Nous utilisons la police proportionnelle *Courier New*, taille 8;
tous les noms de variables et commentaires sont en anglais.

Remarque sur l'utilisation généralisée

Une description détaillée de l'utilisation des scripts, actions et des éditeurs figure au chapitre *4.1 Environnement de développement de scripts dans WinCC.*

3.2.5 Convention : L'interface utilisateur

Généralités

Il est très important de créer l'interface utilisateur avec grand soin. Tous les objets créés dans *Graphics Designer* sont affichés à l'écran dans l'espace de travail de l'utilisateur.
Les vues ainsi créées sont la seule interface entre l'homme et la machine et nécessitent par conséquent beaucoup de soin dans la mesure où elles conditionnent en grande partie la réussite d'un projet. Bien entendu, le fonctionnement de l'installation est plus important que l'aspect de l'écran ; mais à long terme, des vues à la conception négligée peuvent égratigner l'image donnée par un système par ailleurs bien pensé et augmenter les coûts de maintenance.
Ce sont les vues que l'utilisateur (le client) voit tous les jours.

Un système d'écrans n'a pour seule finalité que de fournir à l'utilisateur des informations sur l'état courant de l'installation de process à l'aide des vues. L'interface doit par conséquent fournir des informations **aussi exhaustives et aussi claires que possible**.
WinCC vous permet de configurer librement l'interface utilisateur. Votre configuration dépend du matériel utilisé, des contraintes du process et des conventions existant déjà.

L'opérateur

Vous devez pour définir l'interface utilisateur placer l'opérateur, à qui la configuration est en fin de compte destinée, au centre de vos réflexions.

Si vous parvenez à mettre les informations nécessaires à la disposition de l'opérateur et à les afficher d'une manière claire, ceci se traduira par une **meilleure qualité de la production et moins d'interruptions d'exploitation**. Les travaux de maintenance s'en trouveront réduits.

L'opérateur a besoin d'autant d'informations que possible. Celles-ci lui permettent de prendre les décisions nécessaires pour conduire le process de manière optimale. L'opérateur ne doit pas principalement seulement réagir aux alarmes (dans ce cas, le process est déjà perturbé) mais doit pouvoir anticiper l'évolution du process, à l'aide de son expérience, de sa connaissance du process et des informations fournies par le système de conduite. L'opérateur doit pouvoir prendre des mesures contre une défaillance avant que celle-ci survienne. WinCC vous donne la possibilité de préparer et d'afficher efficacement ces informations à l'intention de l'opérateur.

Combien d'informations doit contenir une vue ?

Savoir combien d'informations doivent être contenues dans une vue dépend de deux aspects d'égale importance :

- Une vue contenant trop d'informations est difficilement lisible et la recherche d'informations demande trop de temps. La probabilité de survenance d'une erreur de conduite augmente.
- Une vue contenant trop peu d'informations augmente la charge de travail de l'opérateur. Il perd alors la vue d'ensemble du process et doit changer fréquemment de vue pour trouver l'information nécessaire. Des réactions et des commandes tardives et l'instabilité du process en sont les conséquences.

Des études montrent qu'un opérateur expérimenté souhaite trouver **dans chaque vue autant d'informations que possible** afin de ne pas devoir changer constamment de vue.
Un débutant par contre est dérouté lorsqu'une vue contient trop d'informations. Il ne trouve pas ou trop tard la bonne information.
Mais nous savons une chose par expérience : **un débutant deviendra vite un opérateur expérimenté alors qu'une personne expérimentée ne redeviendra jamais un novice**.

Masquage d'informations

Les informations à afficher doivent être pertinentes et faciles à comprendre. Vous pouvez **masquer certaines informations (p. ex. numéro de variable, code de point de mesure)** jusqu'à ce que leur affichage devienne nécessaire.

Affichage des informations

Il est recommandé pour l'affichage de valeurs analogiques de combiner des instruments à index à des valeurs numériques. La visualisation graphique (instruments à index, bargraphes, etc.) est mieux adaptée à la perception de l'information.
Afin de prévenir des difficultés pour l'opérateur à reconnaître les couleurs (daltonie), les modifications importantes d'un objet (état) doivent être signalées à la fois par un changement de couleur et par un changement de forme.
Certaines informations doivent sauter aux yeux dans une vue. Ceci suppose l'emploi ciblé de contrastes.

Codage de couleurs

L'oeil perçoit plus rapidement les couleurs que p. ex. le texte. L'utilisation d'un codage de couleurs permet de percevoir plus rapidement l'état dans lequel se trouvent les différents objets. Il est important de toujours bien respecter le code de couleurs choisi. Des conventions de couleur homogènes dans un projet pour représenter les états (p. ex. rouge pour les défaillances) sont maintenant un standard. Les standards d'entreprise existant chez le client doivent être pris en compte.

Affichage de textes

Un certain nombre de règles simples doivent être observées pour améliorer la visibilité du texte.

- La taille du texte doit être adaptée à l'importance de l'information communiquée mais aussi à la distance probable qui séparera l'opérateur de l'écran.
- Utilisez de préférence les minuscules. Elles prennent moins de place et se lisent mieux que les majuscules, bien que ces dernières soient plus facile à reconnaître de loin.
- Un texte horizontal est plus facile à lire qu'un texte vertical ou incliné.
- Utilisez des polices différentes pour des informations différentes (p. ex. pour les noms de point de mesure, notas, etc.).

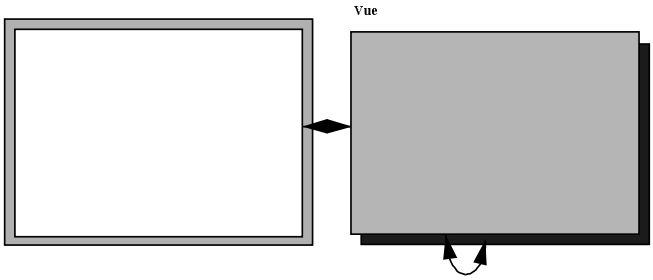
Pas d'entorse au concept

Quel que soit le concept pour lequel vous vous être décidé, vous devez vous y tenir dans la totalité du projet. Vous contribuez ainsi à la possibilité de commander intuitivement les vues de process. Les erreurs de conduite deviennent plus improbables.
Ceci est également le cas pour les objets utilisés. Un moteur ou une pompe doit toujours être représenté de la même façon quelle que soit la vue dans laquelle ces objets sont utilisés.

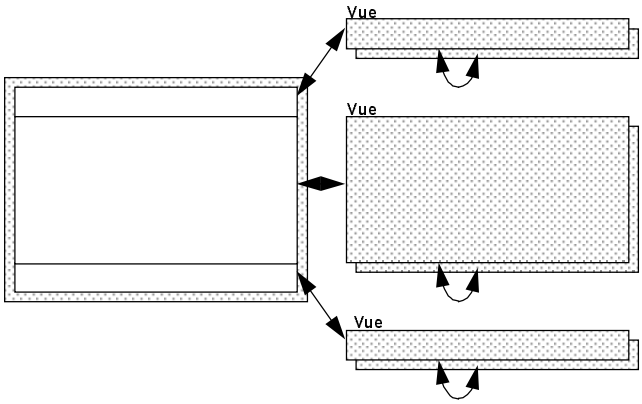
Division de l'écran

Lorsque des écrans de PC standard sont utilisés dans le process, une division de l'écran en trois zones (zone supérieure, zone de travail et zone de boutons) s'est avérée judicieuse.
Cette division de l'écran n'est cependant pas toujours opportune lorsque votre application tourne sur un PC industriel spécifique ou un terminal opérateur (OP).

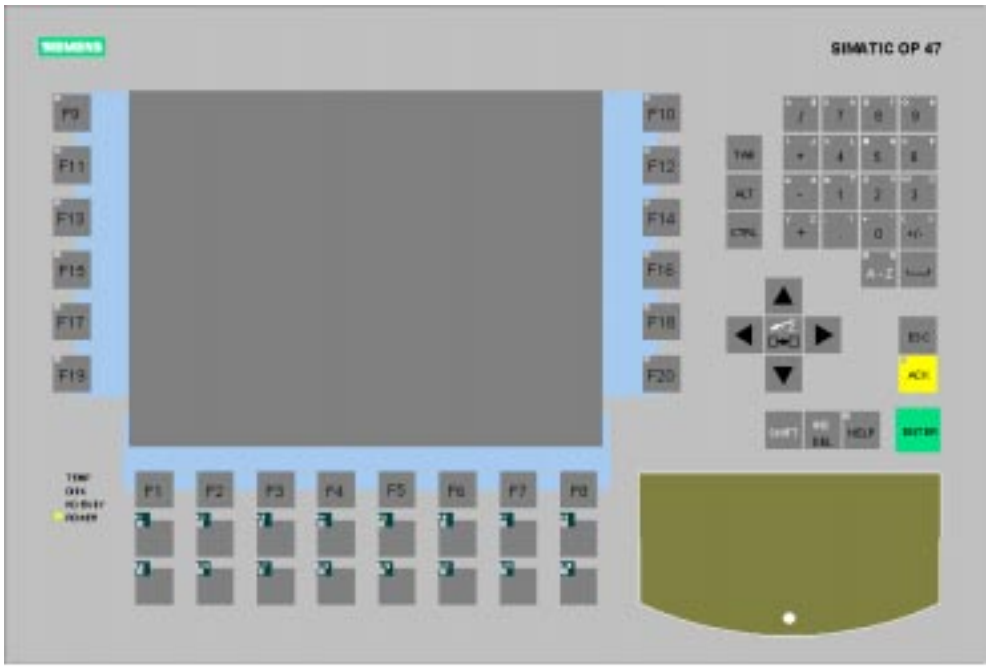
Les vues occupent la totalité de l'écran



L'écran est divisé en trois zones :zone supérieure, zone de boutons et zone de vues de process



Exemple de terminal opérateur



Paramètres / Seuils

Les dimensions des vues peuvent être choisies librement à l'intérieur de certaines limites (mini 1 x 1, maxi 4096 x 4096 pixels). Pour les systèmes monoposte avec moniteur 17", il est recommandé d'utiliser une résolution maximale de 1024 x 768 pixels. Sur les systèmes multiposte (Multi-VGA), une résolution plus grande peut se justifier.
Sur les OP, la résolution est souvent limitée par la technologie utilisée (écran TFT de 640 x 480 à 1024 x 768).

Convention

Les règles suivantes sont applicables aux vues utilisées dans les projets de ce manuel:

Résolution

Nous utilisons dans nos projets des résolutions de 1024 x 768, de 800 x 600 pixels dans des cas exceptionnels. Nous considérons que la palette de couleurs de votre PC doit être paramétrée sur 65536 couleurs au minimum pour garantir une visualisation correcte de nos exemples de projet.

Textes

Nous utilisons pour les désignations de points de mesure la police Courier, pour les textes de simple description et pour tous les autres textes et affichages texte la police Arial. Les polices MS Sans Serif et System sont utilisées pour les boîtes de message de style Windows.
Nous adaptons la taille de police en fonction des nécessités.

Informations dans les vues

Chaque fois que cela paraît souhaitable, nous masquons des informations dans les vues. Nous ne montrons ces informations qu'en cas de nécessité (par conduite opérateur ou déclenchement automatique).
Nous utilisons dans nos projets plusieurs divisions d'écran. Pour de nombreux objets commandés par opérateur, nous formulons des consignes de conduite via l'étiquette.

Division de l'écran

Nous configurerons les possibilités d'édition de l'écran existant en principe. Dans les autres projets, vous utiliserez cependant la division en ligne de titre, zone de travail et zone de boutons.

Remarque sur l'utilisation généralisée

Vous pouvez réutiliser directement dans vos projets la division de base de l'écran.

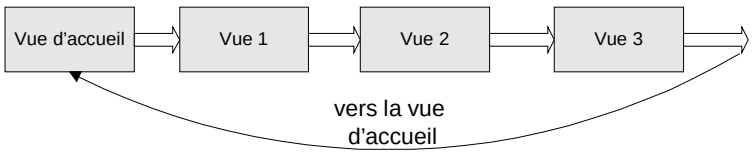
3.2.6 Convention : Concept de conduite

Généralités

Vous disposez pour la conduite de votre application de process des moyens d'introduction connus tels que clavier, souris, écran tactile, joystick industriel. Dans les environnements industriels sévères, dans lesquels l'utilisation d'une souris n'est pas possible, vous pouvez utiliser la commande par **pointeur** et **curseur**. Le pointeur saute entre les champs de commande, le curseur entre les champs de saisie. Toutes les commandes peuvent être protégées contre les accès non autorisés.

Sélection de vue

Le concept de sélection des vues dépend de plusieurs facteurs. L'élément décisif est le nombre de vues et la structure du process à représenter.
Pour les petites applications, les vues peuvent être organisées en tampon cyclique.



Lorsque le nombre des vues est important, une disposition hiérarchique est indispensable à la sélection de vues. Il convient de choisir une structure invariable et simple pour faciliter au personnel de conduite l'apprentissage de la sélection des vues.

La sélection directe de vues est naturellement aussi possible et éventuellement avantageuse pour les très petites applications

(p. ex. chambre frigorifique).

Hiérarchie

Une structure hiérarchisée rend le process plus clair, plus facile à gérer et permet, si nécessaire, d'accéder rapidement à des informations détaillées.
Une hiérarchie à trois niveaux est fréquemment utilisée.

Niveau 1

Le niveau 1 accueille les vues synoptiques.
Ce niveau contient essentiellement des informations sur les sous-systèmes du process et montre les interactions entre ces sous-systèmes.
Ce niveau indique aussi si un événement est survenu à des niveaux inférieurs (alarme).

Niveau 2

Le niveau 2 est celui des vues de process.
Ce niveau contient des informations détaillées sur un sous-système du process et montre les objets de process appartenant à ce sous-ensemble.
Ce niveau indique également l'objet de process concerné par une alarme.

Niveau3

Dans le niveau3 sont rangées les vues de détail.
Ce niveau fournit des informations sur des objets de process, p. ex. régulateurs, vannes, moteurs, etc..
Ce niveau affiche les alarmes, états et valeurs de process. Il fournit également des informations sur l'interaction avec d'autres objets de process.

Convention

La convention suivante sera valable pour tous les projets créés dans le présent manuel:
Nous utiliserons dans nos projets plusieurs concepts de conduite et montrerons leurs différences.

Remarque sur l'utilisation généralisée

Nos projets ne peuvent être considérés que comme des suggestions pour l'élaboration d'un concept de conduite propre à l'utilisateur. En cas d'extension des installations du process, il convient de tenir compte des concepts de conduite existants. Il existe à présent chez de nombreux utilisateurs des standards d'entreprise qu'il convient de prendre en compte dans la configuration.

Nota:
Un concept de conduite prêt à l'emploi est disponible dans l'option WinCC **Basic Process Control**. Ce package optionnel contient en outre d'autres fonctions utiles et performantes (p. ex. stockage).

3.2.7 Convention : Définition des couleurs

Généralités

Les couleurs sont un sujet très discuté en liaison avec le système IHM. WinCC permet de choisir librement les couleurs des traits, cadres, arrière-plans, motifs de remplissage et polices. Toutes les couleurs supportées par Windows sont disponibles. Bien entendu, WinCC permet également de modifier les couleurs et naturellement aussi les autres propriétés des graphiques en runtime.
Une importance particulière revient à la définition des couleurs lorsqu'il s'agit d'obtenir une visualisation claire des process via une configuration économique.

Des couleurs devront toujours être définies pour les domaines ci-dessous. La définition peut éventuellement se faire conformément à DIN *EN 60073*, qui correspond à *VDE 0199*, mais doit avoir été convenue avec l'utilisateur:

- Couleurs des alarmes (arrivées / parties / acquittées)
- Couleurs des états (marche / arrêt / défaut)
- Couleurs des objets de dessin (tuyauteries / niveau de remplissage)
- Couleurs des seuils d'avertissement et seuils d'alarme

Convention

Les règles suivantes sont applicables à toutes les couleurs utilisées dans les projets de ce manuel :
La visualisation correcte des exemples de projet suppose le paramétrage de plus de 256 couleurs.
Dans les exemples de projet, nous utilisons, pour faciliter la perception, une couleur d'arrière-plan spécifique aux thèmes traités (variables, cours de C, configuration de vues). La couleur de fond est plus foncée dans la zone supérieure et la zone de boutons.
Dans le système d'alarmes, un code couleur particulier est affecté à chaque classe d'alarmes et à chaque type d'alarme d'une classe d'alarmes.

Remarque sur l'utilisation généralisée

Vous devez le cas échéant modifier le paramétrage par défaut de WinCC après avoir défini les couleurs.

Vous trouverez un tableau de codage des couleurs dans les *actions C* dans l'annexe chapitre *5.1.6 Table des couleurs*.

3.2.8 Convention : Les cycles de rafraîchissement

Généralités

Il est très important de toujours bien considérer le système global dans l'ensemble lors de la définition des déclencheurs de rafraîchissement. Que faut-il rafraîchir et selon quelle périodicité. Un mauvais choix des temps de cycle de rafraîchissement peut influencer négativement les performances du système IHM.
Il convient en considérant un système global (API - communication - IHM) de détecter les modifications là où elles apparaissent, c'est-à-dire dans le process (API). Dans de nombreux cas, le bus constitue le goulot d'étranglement pour la transmission des données.

Pour définir la périodicité du rafraîchissement des valeurs de mesure, observez à quelle cadence la valeur de mesure est réellement modifiée. Un temps de cycle de rafraîchissement de 500 ms n'aurait aucun sens pour la régulation de température d'une chaudière d'une contenance de 5000l.

Système IHM 32 bits

WinCC est un système IHM 32 bits pur exécutable sous Windows 95 et Windows NT.- Ces systèmes d'exploitation sont optimisés pour la commande événementielle. Si vous respectez ce principe pour la configuration avec WinCC, les problèmes de performances seront plutôt l'exception même si les quantités de données à gérer sont très élevées.

Convention

Les règles suivantes sont applicables pour le rafraîchissement dans les projets du présent manuel:
Le rafraîchissement est effectué si possible par commande événementielle si la tâche d'automatisation le permet. Comme nous travaillons essentiellement avec des valeurs de *variables internes,* nous utilisons souvent les modifications des variables comme déclencheur de rafraîchissement. Lorsque des variables externes sont utilisées, ceci peut entraîner une charge plus importante pour le système. Si la communication permet une transmission déclenchée sur événement, choisir cette option pour les données à temps critique. Les données non critique peuvent alors être appelées par la IHM selon un cycle approprié (procédure d'invitation à émettre).

Remarque sur l'utilisation généralisée

Vous trouverez une description exhaustive de l'utilisation des cycles de rafraîchissement au chapitre *3.3.1 Cycle de rafraîchissement - Où et comment s'effectue le paramétrage*.

3.2.9 Convention : Les droits d'accès

Généralités

Il est nécessaire en exploitation de process que certaines fonctions de conduite soient protégées contre les accès non autorisés. Il convient en outre que seules certaines personnes puissent accéder au système de configuration.
Les utilisateurs et groupes d'utilisateurs peuvent être créés et divers niveaux d'autorisation sont définis dans *User Administrator*. Ces niveaux d'autorisation peuvent être liés aux éléments de commande dans les vues.
Divers niveaux d'autorisation peuvent être affectés individuellement aux utilisateurs et groupes d'utilisateurs.

Convention

Tous les utilisateurs ont le droit d'accès pour la commande dans les exemples de projet *course_00* et *varia_00* .
Dans l'exemple de projet *pictu_00,* la commande n'est possible dans le projet qu'après connexion au système (login). Le mot de passe correspond au nom de projet (*pictu_00*).
Les boutons de sélection des différents thèmes sont liés au niveau d'autorisation *Conduite de projet*.

Remarque pour l'utilisation générale

L'attribution des droits d'accès utilisateur est décrite dans l'exemple de projet *pictu_00* au chapitre *3.3 Terminer WinCC / Autorisation de conduite*.

3.2.10 Convention : Alarmes

Généralités

WinCC supporte deux procédures d'alarmes:

- La **méthode des alarmes déclenchées par changement d'état d'un bit** est une procédure universelle permettant le déclenchement d'alarmes par des automates quelconques. WinCC surveille le changement de front de variables binaires sélectionnées et en dérive des événements d'alarme.
- Les **alarmes déclenchées chronologiquement** supposent que les automates génèrent eux-mêmes les alarmes et les transmettent avec leur horodatage, et éventuellement avec des valeurs de process, à WinCC dans un format prédéfini. Ce système d'alarme permet le classement par ordre chronologique d'alarmes en provenance des divers automates. Voir chapitre *5.2 Documentation du système d'alarmes S5*.

Que faut-il signaler?
Pour convenir des événements et des états à signaler dans les alarmes, on adopte souvent la méthode prétendument sûre qui consiste à signaler tous les événements et tous les changements d'états. C'est alors à l'opérateur de décider des alarmes qu'il veut consulter en premier.
Lorsque trop d'événements sont signalés par un système d'automatisation, l'expérience montre qu'il arrive souvent que des alarmes importantes soient prises en compte trop tard.

Remarque pour l'utilisation générale

La visualisation et la sélection des alarmes pour l'archivage peuvent être modifiées et adaptées aux exigences de l'utilisateur.

3.2.11 Convention : Réalisation

Généralités

Au niveau de la réalisation d'un projet, il est opportun d'utiliser une structure fixe pour mémoriser les données. Le choix du lecteur sur lequel sera mémorisé le projet WinCC est la première étape. L'étape suivante concerne la structure de répertoire, etc.
Notre expérience montre qu'il est conseillé de ranger toutes les données d'un projet dans un répertoire et ses sous-répertoires. Cette démarche présente des avantages pour la réalisation du projet mais avant tout pour la sauvegarde des données.

Nota:
La configuration matérielle et logicielle des PC est très variable. Afin de libérer l'utilisateur de telles contraintes pour l'attribution du lecteur cible au projet, on pourra utiliser des **lecteurs virtuels**. L'affectation du répertoire au lecteur virtuel peut être modifiée à tout moment.

Définition des répertoires

Outre les répertoires créés par WinCC, vous devrez créer, selon les besoins, d'autres répertoires pour Word, Excel et pour les fichiers temporaires.

3.3 Particularités de la configuration avec WinCC

Les prochains chapitres traitent des thèmes généraux concernant la configuration avec WinCC. Ces thèmes constituent un complément à l'aide WinCC.

3.3.1 Cycle de rafraîchissement - Où et comment s'effectue le paramétrage

La définition des cycles de rafraîchissement est l'un des paramétrages les plus importants du système de visualisation. Le paramétrage influe sur les propriétés suivantes :

- La construction de l'image
- Le rafraîchissement des objets de la vue ouverte sur la station de visualisation (*Graphics Designer*)
- L'édition de scripts d'arrière-plan (*Global Script*)
- L'activation du gestionnaire des données et de la communication process

D'autres paramètres temporels se règlent dans le module de traitement de valeurs de mesure (*Tag Logging*) à l'endroit concernant les temps d'archivage.

Gestionnaire de données

Les valeurs courantes des variables sont appelées par le gestionnaire de données, l'instance centrale d'administration de la gestion des variables, en fonction des temps de cycle de rafraîchissement paramétrés. Voir chapitre *3.3.2 Dynamisation WinCC*.

Le gestionnaire de données détermine les nouvelles données de process via les canaux de communication et fournit ces valeurs courantes aux applications. La requête de données signifie ainsi toujours un basculement entre diverses tâches (*Graphics Designer*, *Gestionnaire de données*, etc.). Ceci peut entraîner une charge très variable du système en fonction de la configuration.

3.3.1.1 Rafraîchissement dans une vue

Rafraîchissement dans une vue

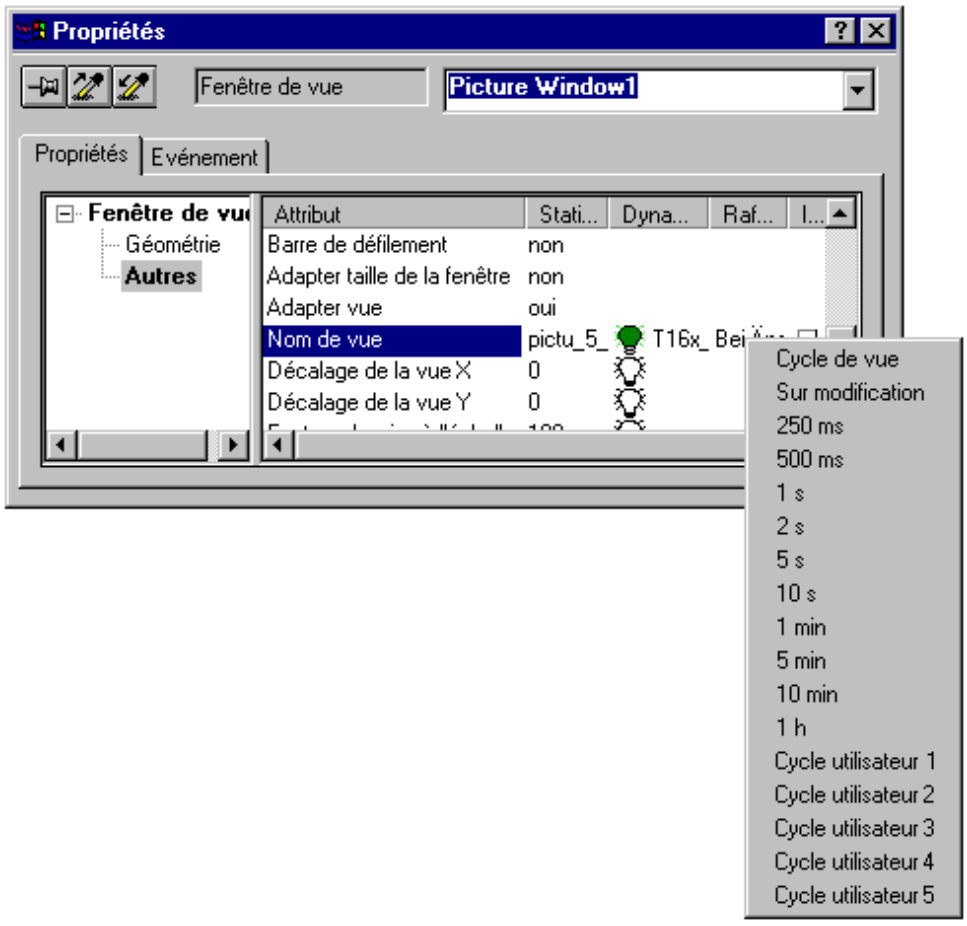
Le rafraîchissement des propriétés des objets dans une vue s'effectue pour les objets dynamisés après l'affichage de la vue. Le temps de cycle de rafraîchissement choisi détermine l'actualité des propriétés de chaque objet dans une vue. Le cycle de rafraîchissement des objets dynamisés peut être paramétré, par le technicien chargé de la configuration ou par le système, pour les modes de dynamisation suivants:

Mode de dynamisation	Paramétrage standard	Adaptation de la configuration
Dialogue de configuration	Déclencheur sur variable 2 s Ou déclencheur sur événement (p. ex. commande)	Adaptation des cycles temporels
Assistant Dynamic-Wizard	Possibilité de sélection, en fonction du type de dynamisation, entre • déclencheur sur événement, • Cycle temporel • Déclencheur sur variable	Adaptation des cycles temporels, événements ou variables
Liaisons directes	déclencheur sur événement	
Liaison de variable	Déclencheur sur variable 2 s	Adaptation des cycles temporels

Mode de dynamisation	Paramétrage standard	Adaptation de la configuration
Dialogue de dynamisation	Déclencheur sur variable 2 s	Adaptation des cycles temporels, déclencheurs sur variable
Action C pour propriétés	Cycle temporel 2 s	Adaptation des cycles temporels, déclencheurs sur variable Lecture directe dans API
Propriété d'objet	Paramétrage en fonction de la dynamisation	Edition de la colonne Cycle de rafraîchissement

Les cycles de rafraîchissement disponibles pour la sélection sont proposés par WinCC et peuvent être complétés par des cycles définis par l'utilisateur.

Sélection des cycles de rafraîchissement, p. ex. pour la propriété d'un objet:



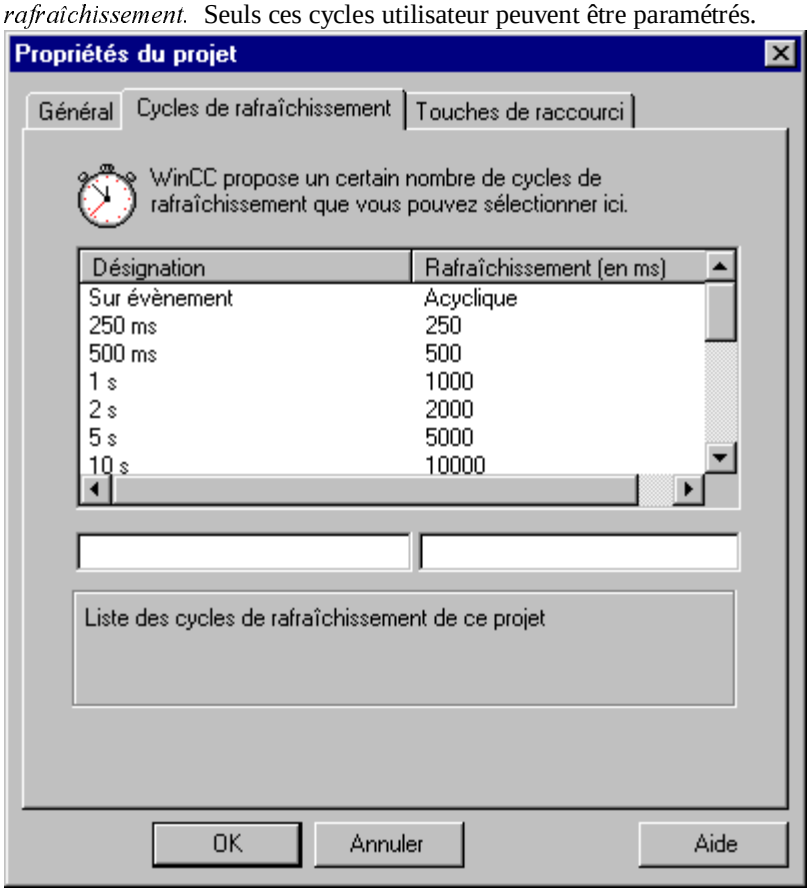
3.3.1.2 Types de cycle de rafraîchissement

On distingue les types de cycle de rafraîchissement suivants:

Type	Paramétrage standard
Cycle standard	Temps de cycle 2 secondes
Temps de cycle	2 secondes
Déclencheur sur variable	2 secondes
Cycle de vue	2 secondes
Cycle de fenêtre	Sur modification
Cycles définis par l'utilisateur	Cycle utilisateur 1 :2 s Cycle utilisateur2 :3 s Cycle utilisateur3 :4 s Cycle utilisateur4 :5 s Cycle utilisateur5 :10 s

Cycle utilisateur

Il est possible de définir jusqu'à 5 cycles utilisateur **spécifiques à un projet**. Si, dans le *Control Center*, vous avez sélectionné le nom de projet dans la structure arborescente gauche, le bouton  de la barre d'outils permet d'ouvrir la boîte de dialogue *Propriétés de projet*. Les cycles utilisateur 1...5 sont proposés, pour des définitions spécifiques à un projet, à la fin de la liste des temps de cycle de rafraîchissement paramétrés dans la boîte de dialogue *Propriétés de projet*, onglet *Cycle de rafraîchissement*. Seuls ces cycles utilisateur peuvent être paramétrés.



Ceci permet de définir des temps de cycle pas encore disponibles comme périodicités de rafraîchissement (p. ex. 200 ms).
Le temps de cycle utilisateur peut être défini entre 100 ms et 10 h. La désignation des cycles utilisateur peut être choisie librement.

Ces temps de cycle spécifiques à un projet peuvent être utilisés pour des objets sélectionnés dont le temps de cycle de rafraîchissement doit être modifié ultérieurement. Une optimisation pourrait être, par exemple, un motif de modification des temps de cycle. Les cycles de rafraîchissement définis par l'utilisateur permettent de modifier le temps de cycle paramétré ultérieurement de manière centralisée en **un** seul endroit. Il n'est alors plus nécessaire d'adapter les divers objets des vues. Cette méthode de définition des temps de cycle utilisateur est donc à prendre en considération en vue d'obtenir des projets **à grande facilité de maintenance**.

3.3.1.3 Signification des cycles de rafraîchissement

Avant d'utiliser les différents cycles de rafraîchissement possibles, il convient d'en examiner tout d'abord la signification.

On distingue les types de cycle de rafraîchissement suivants:

Type	**Signification**
Cycle standard	Temps de cycle
Temps de cycle	La propriété ou l'action de l'objet est rafraîchie selon la périodicité paramétrée. Cela signifie que les variables sont demandées **individuellement** à chaque requête par le gestionnaire de données.
Déclencheur sur variable	Le contenu des variables est déterminé par le système, qui vérifie si leur valeur a été modifiée, dans la périodicité paramétrée. Toute modification d'au moins une variable sélectionnée, dans la période paramétrée, déclenche les propriétés ou actions dépendant de ce déclencheur. **Toutes** les valeurs de variable sont demandées **ensemble** par le gestionnaire de données.
Cycle de vue	Rafraîchissement des propriétés de l'objet de vue courant et de tous les objets déclenchés par le déclencheur de rafraîchissement 'Cycle de vue'.
Cycle de fenêtre	Rafraîchissement des propriétés de l'objet fenêtre et de tous les objets ayant comme cycle de rafraîchissement le cycle de fenêtre.
Cycles définis par l'utilisateur	Temps de cycle pouvant être défini de manière spécifique à un projet.
Action C pour lecture directe dans API	Des fonctions internes contenues dans les *actions C* permettent de lire directement des valeurs dans les API. L'exécution des instructions suivantes de *l'action C* n'est reprise qu'après la lecture des valeurs de process (lecture synchrone).

Nota:
La requête de communication de la valeur réelle courante d'une variable par le gestionnaire de données entraîne toujours un basculement de tâche et un échange de données entre les tâches. Le gestionnaire doit en outre émettre une requête de communication des valeurs de variables par le canal de communication des automates connectés. Ceci est réalisé en fonction du type de communication par des télégrammes de requête envoyés à l'interface de communication (FETCH) et des télégrammes de données émis en retour par l'automate à WinCC.

3.3.1.4 Remarques sur l'utilisation des cycles de rafraîchissement

Le paramétrage suivant est recommandé pour l'utilisation des cycles de rafraîchissement en fonction du mode de rafraîchissement utilisé:

Type	Paramétrage standard	Recommandation pour la configuration
Cycle standard	Temps de cycle 2 secondes	*Dialogue de dynamisation* ou *action C*: Utilisez impérativement un déclenchement par variable lorsqu'il existe dans le process une dépendance par rapport à des variables. Ceci permet de réduire le nombre des changements de tâche et le volume de la communication entre les tâches. Le déclenchement par variable *sur modification* ne doit être utilisé qu'avec discernement car il entraîne une charge plus lourde pour le système ! Dans ce cas, le système vérifie constamment s'il y a eu modification de valeurs de variable. Ce système d'invitation à transmettre augmente toujours la charge du système. Pour des objets standard, il est recommandé d'utiliser un temps de cycle de 1 à 2 secondes.
Temps de cycle	2 secondes	Choisissez toujours le temps de cycle en fonction du type d'objet ou de la propriété de l'objet. Tenez compte de l'inertie des composants du process (remplissage de réservoirs ou températures par opposition aux manoeuvres de commutation). Pour des objets standard, il est recommandé d'utiliser un temps de cycle de 1 à 2 secondes.
Déclencheurs sur variable	2 secondes (*dialogue de dynamisation*)	Utilisez **de préférence** ce mode de rafraîchissement lorsqu'il est configurable (fonction du mode de dynamisation) ! En cas de dépendance par rapport à des variables, prendre impérativement en compte **toutes** les variables responsables d'une modification de la propriété ou de l'exécution de l'action. Seules les variables figurant dans la liste peuvent peuvent servir de déclencheur pour rafraîchir la propriété ou l'option dynamisée. N'employez le **déclenchement par variable** *sur modification* que dans des cas bien précis. Le déclencheur de cette propriété ou action sera activé dès que la valeur d'une des variables sélectionnée sera modifiée. Ce mécanisme d'invitation à transmettre entraîne une plus grande charge pour le système.
Cycle de vue	2 secondes	Ne choisissez un cycle plus court que si les propriétés de l'objet vue dynamisées sont elles-mêmes modifiées dans une période plus courte et doivent par conséquent être rafraîchies. L'augmentation de ce temps de cycle de vue réduit la charge du système.
Cycle de fenêtre	Sur modification	Ce paramétrage est approprié lorsqu'il s'agit d'une fenêtre de vue affichée p. ex. pour la modification de grandeurs de process (boîte de process). Paramétrer le rafraîchissement de la fenêtre et de son contenu sur un déclencheur sur variable ou sur un temps de cycle si la fenêtre de la vue est affichée constamment pour information (p. ex. structuration de l'écran).

Type	Paramétrage standard	Recommandation pour la configuration
Action C pour lecture directe dans API		Les fonctions internes (p. ex. GetTagWordWait) servant à la lecture synchrone de valeurs de process (directement dans l'API) ne doivent être utilisées que judicieusement. L'emploi de ces fonctions nécessite une invitation à transmettre par le système (pilotage d'action) qui entraîne une charge de communication plus élevée.

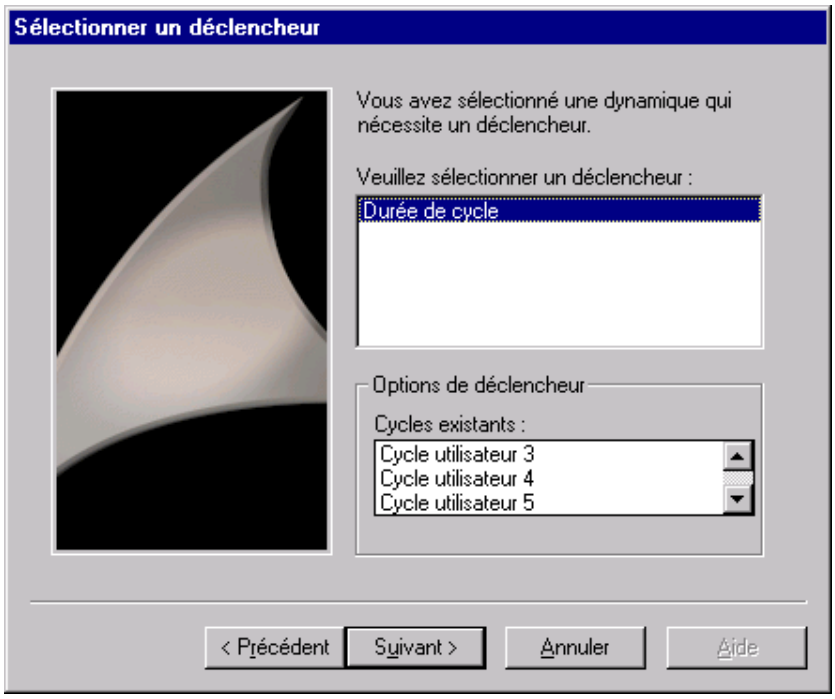
Les exemples ci-dessous montrent où paramétrer les cycles de rafraîchissement :

Dialogue de configuration

Ce dialogue est affiché à la configuration d'un *objet complexe* →*Champ d'E/S*. Mais il peut également être ouvert par un **D** sur l'objet correspondant.

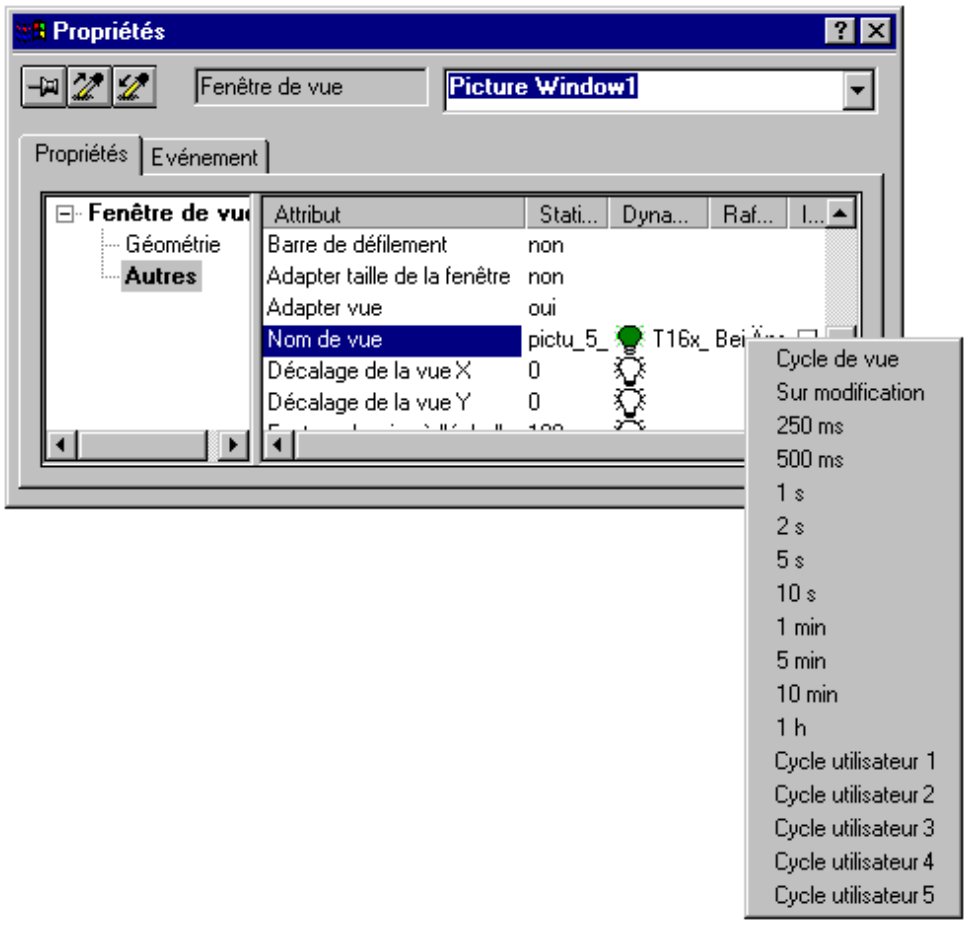
Dynamic Wizard

Cette page est affichée à la sélection de *Dynamiser une propriété* de l'onglet *Dynamiques Standard* dans le *Dynamic-Wizard* avec **GG**.



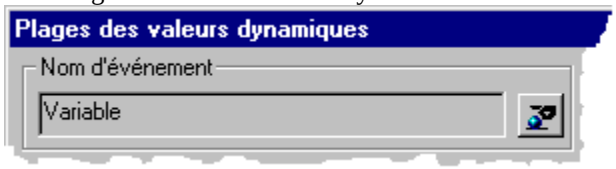
Liaison de variables pour propriété d'objet

Le menu ci-dessus est affiché à la sélection de la colonne *'Rafr.'* avec **D** pour une propriété d'objet dynamisée par une variable .



Dialogue de dynamisation

Dans les *dialogues de dynamisation*, les dialogues sont appelés par sélection du bouton déclencheur du dialogue de modification du cycle de rafraîchissement.



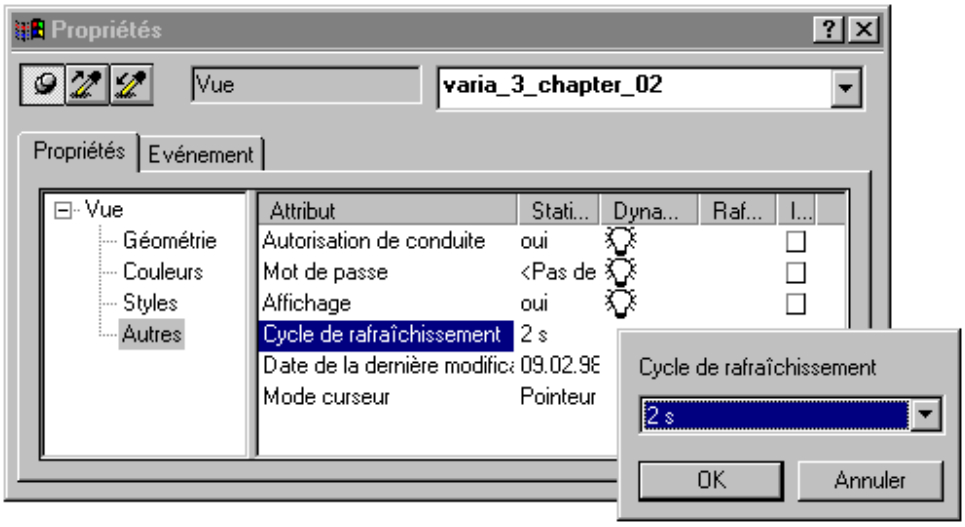
Action C pour propriété

Dans une *action C*, le dialogue de modification du cycle de rafraîchissement est appelé dans l'éditeur par sélection du bouton déclencheur.

Les cycles de rafraîchissement paramétrés par défaut peuvent être modifiés de la manière suivante:

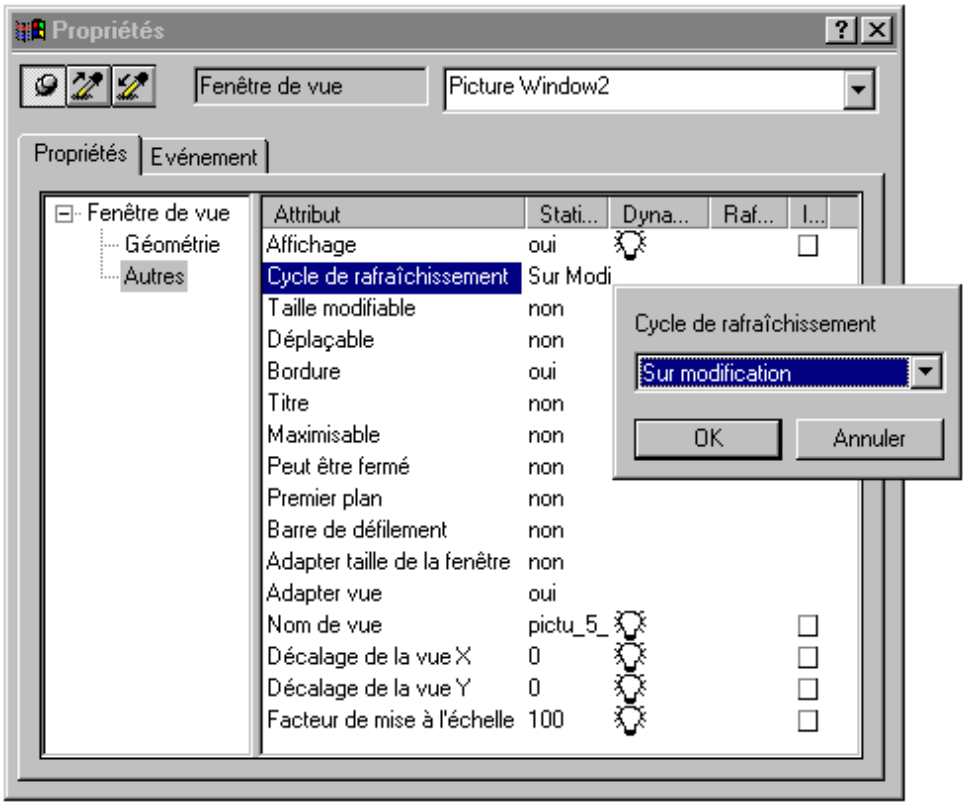
Cycle de vue

Modification du cycle de vue



Cycle de fenêtre

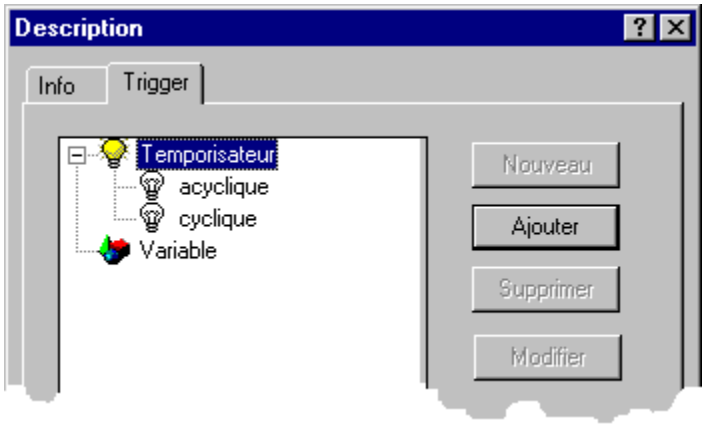
Modification du cycle de fenêtre



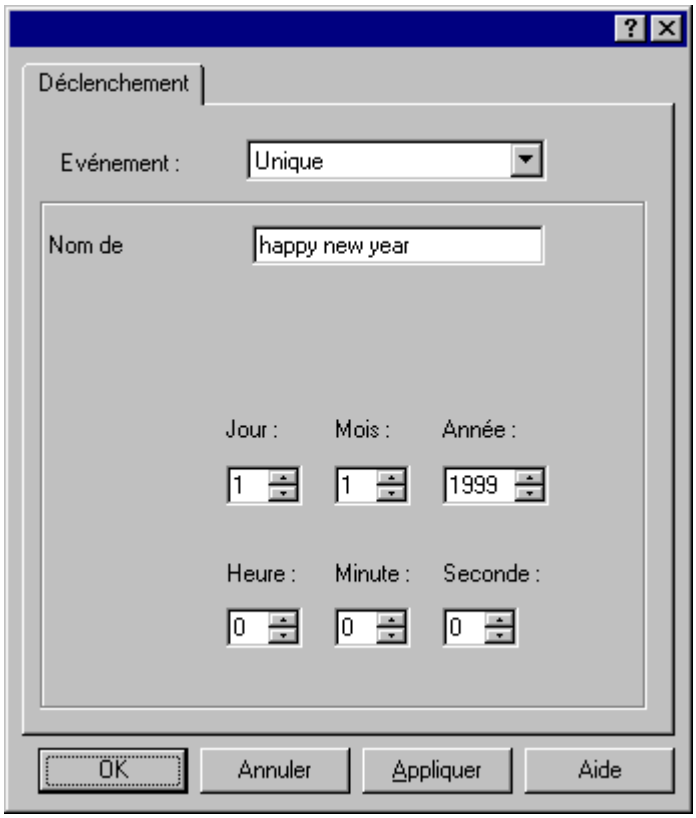
3.3.1.5 Exécution de scripts en arrière-plan (Global Script)

L'exécution de scripts en arrière-plan (*Global Script*) dépend de plusieurs grandeurs fonction de la configuration:

- Déclencheur temporel (exécution cyclique, exception : acyclique = une seule fois)
  - Temps de cycle
  - Instant
- Déclencheur sur événement



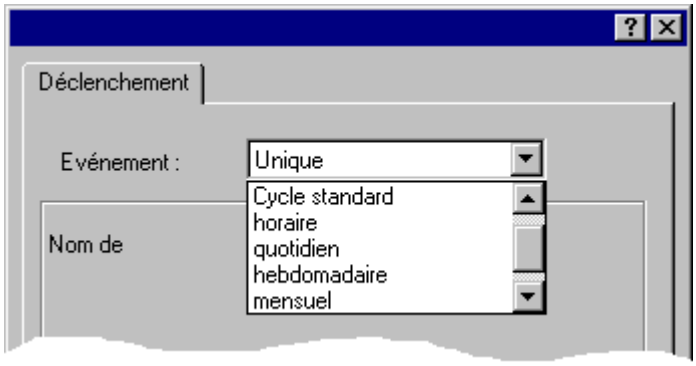
ou une seule fois



Temps de cycle

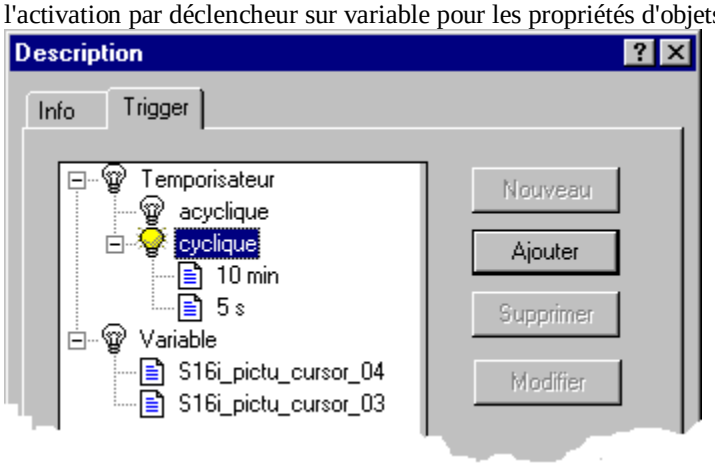
La périodicité configurée pour l'action globale détermine **quand** les instructions constituant l'action définie doivent être exécutées. Outre le cycle standard déjà décrit et les paramétrages de temps de cycle correspondants de 250 ms à 1 h (ou cycle utilisateur 1 à 5), il est également possible de sélectionner les déclencheurs temporels :

- Heure (minute et seconde)
- Jour (heure, minute, seconde)
- Semaine (jour, heure, minute, seconde)
- Mois (jour, heure, minute, seconde)
- Année (mois, jour, heure, minute, seconde)

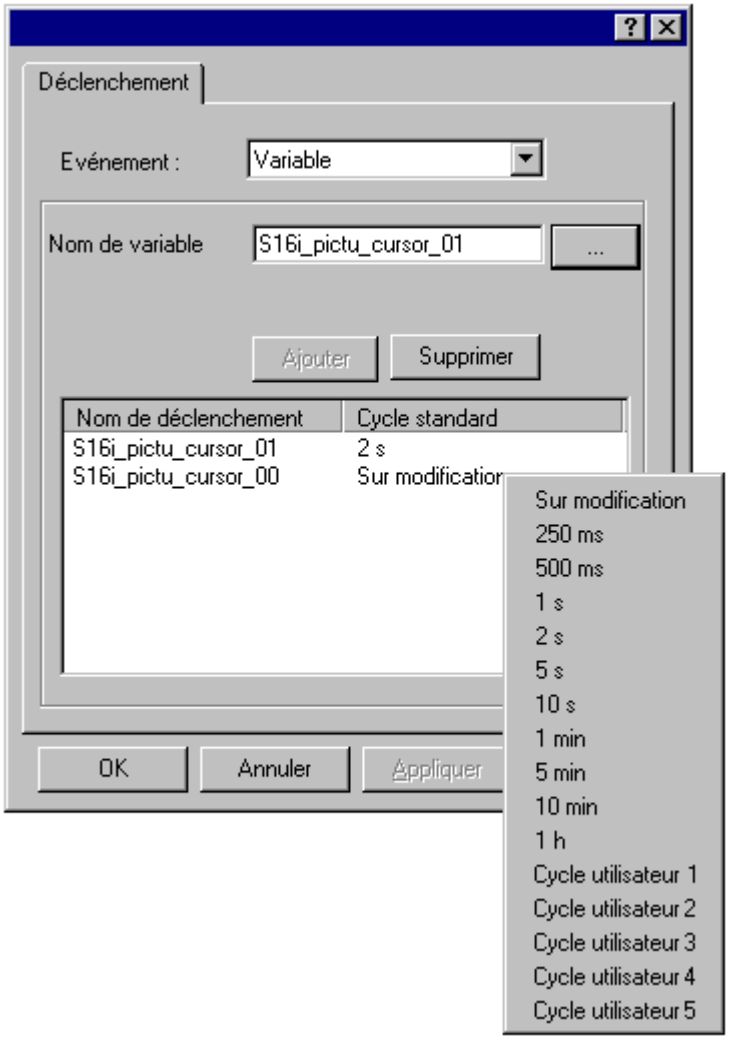


Déclencheur sur variable

Le déclencheur sur événement doit être paramétré comme déclencheur sur variable si l'action est activée en fonction de la valeur d'une ou de plusieurs variables. Ceci s'effectue de manière analogue à l'activation par déclencheur sur variable pour les propriétés d'objets.



Le temps de cycle paramétré par défaut est 2 secondes. Le technicien chargé de la configuration peut paramétrer les périodicités ci-dessous à la place de la valeur par défaut:



La valeur de la variable sélectionnée est déterminée au début et à la fin de chaque période paramétrée. Le déclencheur est déclenché pour l'action globale lorsque la valeur d'une variable au moins a été modifiée.
N'oubliez pas que le déclenchement *sur modification* de valeurs de variables augmente considérablement la charge du système. Ce paramétrage n'est pas toujours opportun. Les remarques faites pour le rafraîchissement d'objets sont également valables ici.

Toutes les actions que vous avez définies comme actions globales **ne sont pas testées et activées spécifiquement à l'objet**, mais uniquement en fonction des temps de cycle paramétrés ou des déclencheurs événementiels. Utilisez par conséquent les actions globales très judicieusement et évitez les instructions superflues dans les actions afin de ne pas surcharger inutilement le système. N'utilisez pas trop de temps de cycle et trop de temps de cycle courts pour l'exécution des actions.

3.3.2 Dynamisation WinCC

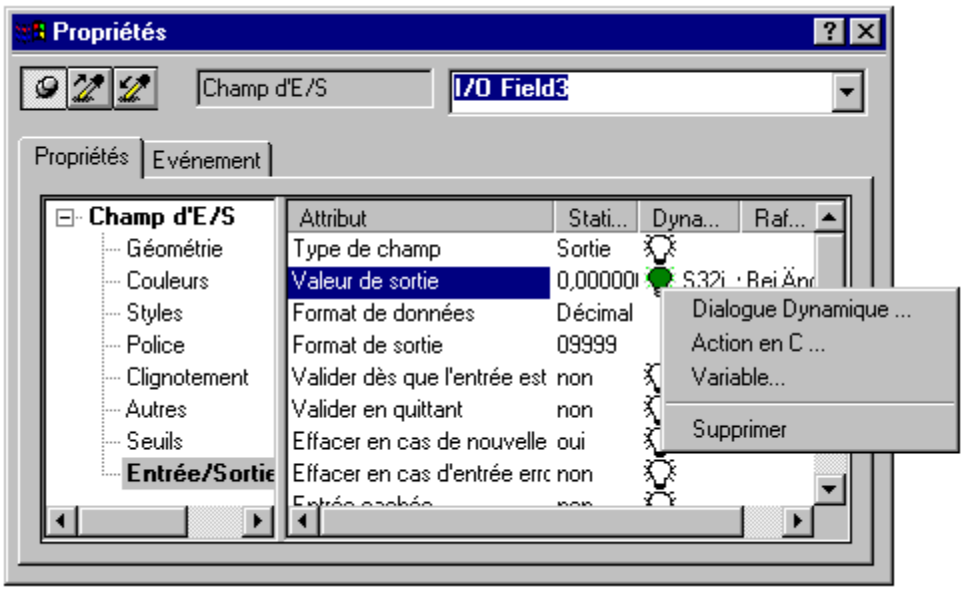
Définition

Nous entendons par dynamisation le changement d'états (p. ex. position, couleur, police, etc.) et la réaction à des événements (p. ex. clic de souris, commandes clavier, modification de valeurs, etc.) au runtime.

Chaque élément d'une fenêtre est considéré comme un objet autonome. La fenêtre est elle-même un objet du type *objet vue*.
Dans le *Système graphique WinCC*, chaque objet possède des *propriétés* et des *événements*. A quelques rares exceptions près, ceux-ci peuvent être dynamisés. Ces exceptions concernent pour l'essentiel les *propriétés* et *événements* qui ne produisent pas d'effet au runtime. Ils ne possèdent pas de symbole de propriété ou d'événement dynamisable.

3.3.2.1 Dynamisation de propriétés

Les propriétés d'un objet (position, couleur, police, etc.) peuvent être activées statiquement et modifiées dynamiquement au runtime.
Toutes les propriétés marquées d'une ampoule dans la colonne *Dynamisation* sont dynamisables. Lorsqu'une propriété est dynamisée, un symbole de couleur est affiché à la place de l'ampoule blanche en fonction du type de dynamisation. Les thèmes (p. ex. géométrie) dynamisés sont affichés en gras.



3.3.2.2 Dynamisation d'événements

Les événements d'un objet (p. ex. clic de souris, commandes clavier, modification de valeurs, etc.) peuvent être testés au runtime et exploités dynamiquement.
Tous les événements marqués d'une flèche dans la colonne *Action* sont dynamisables. Lorsqu'un événement est dynamisé, une flèche de couleur est affichée à la place de la flèche blanche en fonction du mode de dynamisation. Les thèmes (p. ex. divers) dynamisés sont affichés en gras.

3.3.2.3 Modes de dynamisation des objets

Les objets d'une vue de process peuvent être dynamisés de diverses manières. Les divers dialogues standard de dynamisation sont orientés vers divers domaines d'application et entraînent en partie aussi des résultats différents.

Vue d'ensemble

Mode de dynamisation	A	B	Avantage	Inconvénient
Dynamic Wizard	x	x	Démarche guidée standard pour la configuration	Uniquement pour certaines dynamisations. Génère toujours une *action C* !
Liaisons directes		x	La dynamisation la **plus rapide** dans une vue ; performances **maximales** en runtime	Utilisable uniquement pour une **seule** liaison et uniquement dans une vue.
Liaison de variables	x		Facile à configurer	Possibilités de dynamisation restreintes
Dialogue de dynamisation	x		Rapide et clair ; pour plages de valeurs ou alternatives multiples ; bonnes performances au runtime	Pas utilisable pour toutes les dynamisations
Action C	x	x	Possibilités presque illimitées de dynamisation	Risques d'erreur dus à des instructions

Mode de dynamisation	A	B	Avantage	Inconvénient
			grâce au langage de scripts puissant (C Ansi)	C erronées moindre performance par rapport aux autres modes de dynamisation ; toujours s'assurer que l'objectif ne peut pas être atteint par un autre mode de dynamisation.

Légende:

A Dynamisation de la propriété d'objet
B Dynamisation de l'événement d'objet

Appel des dialogues pour la dynamisation

Dialogue	Appel
Dialogue de configuration	Tous les objets ne possèdent pas un tel dialogue. Automatique à la création de ces objets. *Sélectionner objet dans vue →, appuyer sur la touche MAJ et la maintenir enfoncée →* **GG**. *Sélectionner objet dans vue →* **D**, *ouvrir le menu contextuel →, lancer dialogue de configuration*
Dynamic Wizard	*Sélectionner objet dans vue → Sélectionner propriété ou événement → Sélectionner assistant et lancer avec* **GG**. L'assistant *Dynamic Wizard* se sélectionne via les commandes *Affichage → Barres d'outils...* .
Liaison directe	*Sélectionner objet dans vue → Afficher Propriétés d'objet → Sélectionner onglet Evénement → Ouvrir le menu contextuel avec* **_GG** *dans la colonne action → Sélectionner liaison directe.*
Liaison de variables	*Sélectionner objet dans vue → Afficher propriétés d'objet → Sélectionner l'onglet Propriétés → Ouvrir le menu contextuel dans la colonne Dynamisation avec* **D** *→ Sélectionner variable → Sélectionner et valider la variable correspondante dans le dialogue.*
Dialogue de dynamisation	*Sélectionner objet dans vue → Afficher Propriétés d'objet → Sélectionner onglet Evénement → Ouvrir le menu contextuel avec* **D** *dans la colonne dynamisation → Sélectionner dialogue de dynamisation → Configurer et valider la dynamisation correspondante dans le dialogue.*
Action C	*Sélectionner objet dans vue → Afficher Propriétés d'objet → Sélectionner onglet Propriétés ou Evénement → Ouvrir le menu contextuel avec* **D** *dans la colonne Dynamisation ou Action → Sélectionner action C → Configurer et traduire l'action C correspondante.*

Résultats et visualisation

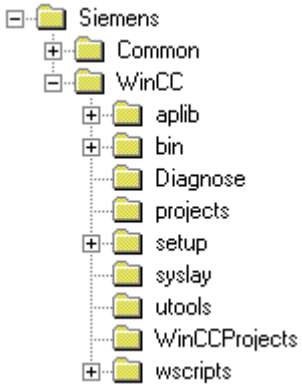
Dialogue	Résultat	Visualisation
Dynamic Wizard	Il y a toujours création d'une *action C*.	*Flèche verte*
Liaison directe		*Flèche bleue*
Liaison de variables		*Ampoule verte*
Dialogue de dynamisation	*Action C* (InProc) générée automatiquement ; cette *action C* peut être ensuite étendue, mais l'avantage des meilleures performances est alors perdu.	*Flèche rouge* Changement en *flèche verte sur modification dans l'action C.*
Action C	Script C configuré	*Flèche verte* *Flèche jaune* - l'action doit être encore traduite

3.3.3 Environnement système WinCC

WinCC est chargé dans le chemin d'installation standard *C:\Siemens\WinCC*. Le chemin standard peut être modifié pendant l'installation.

3.3.3.1 Structure de répertoires du système WinCC

La structure du répertoire, sans les options et sans les exemples, est construite de la manière suivante.



Fichiers dans le répertoire standard WinCC

Dans le chemin de WinCC standard, les répertoires et fichiers suivants intéressent le responsable de la configuration et de la mise en service.

Répertoire	Nom de fichier, extension	Remarque
Diagnostic	License.log	Entrées courantes dans le journal concernant les vérifications et violations de licence
	License.bak	Fichier journal des informations de licence du dernier démarrage
	WinCC_-Op_01.log	Messages opérateur générés par WinCC au runtime.
	WinCC_-Sstart_01.log	Messages système générés par WinCC au démarrage. Fichier très important pour la **recherche d'erreurs**. Le fichier contient des messages sur les variables manquantes, les scripts mal exécutés.
	WinCC_-Sys_01.log	Messages système générés par WinCC au runtime. Fichier très important pour la **recherche d'erreurs**. Le fichier contient des messages sur les variables manquantes, les scripts mal exécutés.
	S7chn01.log	Messages système du canal utilisé (en l'occurrence S7)
aplib	Chemin de bibliothèque	Fichiers d'entête ; toutes les fonctions standard et toutes les fonctions internes sont rangées dans des sous-répertoires.

Fichiers dans le répertoire standard WinCC

Dans le chemin WinCC standard, les fonctions et symboles valables pour tous les projets sont rangés dans les répertoires suivants:

Répertoire	**Sous-répertoire, nom de fichier**	**Remarque**
aplib	library.pxl	Symboles de la bibliothèque standard de WinCC
	Report, Wincc, Windows	Répertoires des fonctions standard ; celles-ci peuvent être adaptées à **tout moment**.
	Allocate, C_bib, Graphics, Tag	Répertoires des fonctions internes ; **ne peuvent pas** être adaptés.
syslay		Toutes les mises en page d'impression copiées automatiquement par WinCC dans le chemin de projet dans le répertoire *prt*, à la création d'un projet.
wscripts	Dynwiz.cwd	*Assistant Dynamic Wizard* de *Graphics Designer.* Possibilités de créer ses propres scripts à tout moment. Ces scripts ont toujours l'extension **.wnf**
	wscripts.deu	Ce chemin contient les fichiers de scripts pour l'*allemand*. Ce chemin est fonction de la langue installée.
	Wscripts.enu	Ce chemin contient les fichiers de scripts pour l'*anglais*. L'anglais étant la langue par défaut, ce chemin est toujours créé.
	Wscripts.fra	Ce chemin contient les fichiers de scripts pour le *français*. Ce chemin est fonction de la langue installée.

Fichiers dans le répertoire standard WinCC

Les programmes utilisateur ci-dessous sont rangés dans les répertoires ci-après lors de l'installation de WinCC :

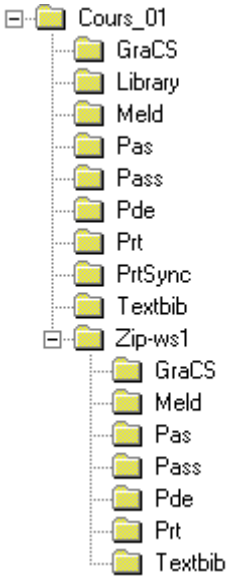
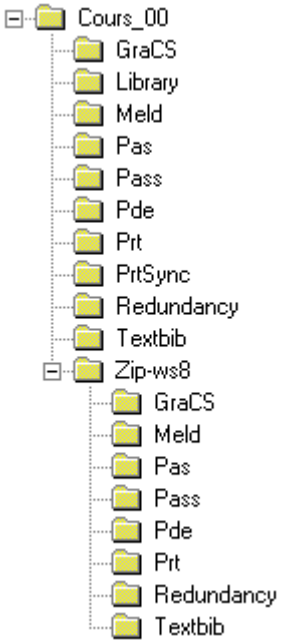
Répertoire\ Fichier	Remarque
\sqlany\isql.exe	Programme interactif permettant de consulter une base de données d'un projet WinCC.
\bin\Wunload.exe	Assistant (Wizard) servant à vider les tableaux en ligne dans la base de données du projet WinCC, p. ex. suppression des alarmes et données de points de mesure mémorisées. L'assistant paramètre automatiquement les tables de runtime pour le déchargement ; l'utilisateur peut cependant ajouter ou supprimer d'autres tables dans la liste à tout moment. Cet outil doit être utilisé hors ligne pour un projet WinCC. Cet outil n'est pas utilisable en mode runtime. L'exportation des alarmes et des valeurs de mesure au runtime est possible avec le package optionnel *STORAGE*.
\bin\Wrebuild.exe	Assistant (Wizard) pour la restauration de la base de données ; **ne peut pas** être utilisé en runtime!
\SmartTools\CC_GrafikTools\metaVw.exe	Composant de visualisation de fichiers graphiques (p. ex. travaux d'impression, symboles exportés) au format EMF (extended meta file).
\SmartTools\CC_GrafikTools\wmfdcode.exe	Composant de visualisation de fichiers graphiques au format WMF (windows meta file).
\SmartTools\CC_OCX_REG\ocxreg.exe	Pour enregistrement ou suppression d'autres composants OLE (OCX).
\SmartTools\CC_OCX_REG\Regsvr32.exe	appelé par ocxreg.exe.

3.3.4 Environnement projet WinCC

Nota:
Créez pour vos projets WinCC un répertoire de projets spécifique, p. ex. WinCC_Projets. Ceci permet d'obtenir une séparation claire entre le système WinCC et les données configurées. La sauvegarde des données s'en trouve facilitée. Le risque de perte de données (par erreur de manipulation) lors d'une désinstallation de WinCC est très largement évité.

3.3.4.1 Projet WinCC - Structure de répertoires

Un projet WinCC est constitué d'une structure globale de répertoires et de leur contenu. Lorsqu'un projet est créé dans *Control-Center* (via *Fichier →Nouveau*), une nouvelle structure de répertoires est constituée de la manière suivante :

WinCC Standard	WinCC avec options
	

Contenu des répertoires de projet

Répertoire	Extension	Remarque
Chemin de projet	.db	Base de données contenant les données de configuration
	rt.db	Base de données contenant les données de runtime, valeurs de mesure, alarmes
	.mcp (master control program)	Fichier principal du projet WinCC. Ce fichier ouvre le projet.
	.pin	Projet.pin
GraCS	.pdl (picture design language)	Les vues configurées
	.sav	Fichiers de sauvegarde des fichiers de vue avec l'état de

Répertoire	Extension	Remarque
		configuration le plus récent
	.bmp (bitmap), .wmf (windows meta file), .emf (extended meta file)	Fichiers de vue
	.act (action)	*Actions C* exportées
	.pdd	*Default.pdd* Paramètre de l'éditeur graphique (paramétrage par défaut des objets dans la palette d'objets)
Library	.h (header)	*Ap_pbib.h* (déclarations de fonctions de projet)
	.pxl	*Library.pxl* (bibliothèque de symboles de projet)
	.fct	fonctions de projet)
	.dll (dynamic link library)	Bibliothèques de fonctions utilisateur créées avec un environnement de développement C.
Meld		
Pas	.pas (définition d'actions)	Actions de projet exécutées en arrière-plan en fonction du déclencheur paramétré
Pass		
Pde		
Prt	.rpl (report picture language) .rp1 (...modèle de ligne)	Mises en page pour travaux d'impression Les mises en page standard WinCC prédéfinies commencent toujours par @. Toutes les grandeurs du système (également variables) sont caractérisées par ce préfixe.
Nom d'ordinateur p. ex. Zip-ws1	\GraCS\GraCS.ini	Fichier d'initialisation de l'éditeur graphique

Option : Fichiers pouvant être créés pendant la configuration

Répertoire	Extension	Remarque
en partie librement définissable	.ini	Fichier d'initialisation du simulateur avec informations pour appel.
	.sim	Variables internes avec paramétrage pour la simulation
	.csv	Textes exportés de la bibliothèque de textes
	.txt	Alarmes exportées du système d'alarmes (*Alarm Logging*)
	.emf	Travaux d'impression copiant les données d'impression dans un fichier
	.log	Fichiers Log
	.xls .doc .wri	Fichiers créés avec d'autres applications mais utilisés dans un projet WinCC

3.3.5 Lancement automatique de projet par WinCC

Condition

Le système IHM (WinCC) installé doit être lancé automatiquement au démarrage du système Windows. L'opérateur travaillant sur la station IHM n'a besoin d'aucune connaissance de la manipulation de Windows (p. ex. appel de WinCC sous Windows 95 ou sous Windows NT).

Solution

WinCC est lancé automatiquement à l'allumage du PC par la routine de démarrage. Ce paramétrage s'effectue dans le dossier *Démarrage* de Windows.

Création de liaisons

Etape	Procédure pour NT4.0 :
1	Dans l'explorateur Windows, passez dans le répertoire *WinNT\Profiles\All Users\Démarrer\Programmes\Démarrage*. WinNT est le répertoire dans lequel Windows NT a été installé
2	Créez dans le dossier une nouvelle liaison en activant ☞D →*Nouveau* → *Liaison*
3	Créez la liaison au programme *mcp.exe* (Master Control Program) dans le répertoire *\bin* de WinCC.
4	Attribuez un nom à la liaison.

Le module *Control Center* de WinCC est alors automatiquement lancé. WinCC est automatiquement lancé avec le projet dernièrement édité ou activé.
Pour démarrer un système en mode runtime, le projet doit donc avoir été quitté alors qu'il était activé.

Nota:
Si la combinaison de touches *CTRL + MAJ* n'est pas bien verrouillée, WinCC démarre en mode configuration, même si le projet a été quitté à l'état activé, lorsqu'on appuie sur cette combinaison de touches au démarrage de WinCC.

L'opérateur voit alors la vue de démarrage du système qui lui est familière. Pour que l'opérateur ne puisse pas, par inadvertance ou intentionnellement, basculer en configuration (exécutée en arrière-plan) ni utiliser des applications Windows qu'il n'est pas habilité à manipuler, des mesures appropriées doivent être prises. L'opérateur ne doit pas non plus accéder à la fenêtre de runtime de la base de données, car il peut ainsi terminer la connexion à la base de données WinCC.

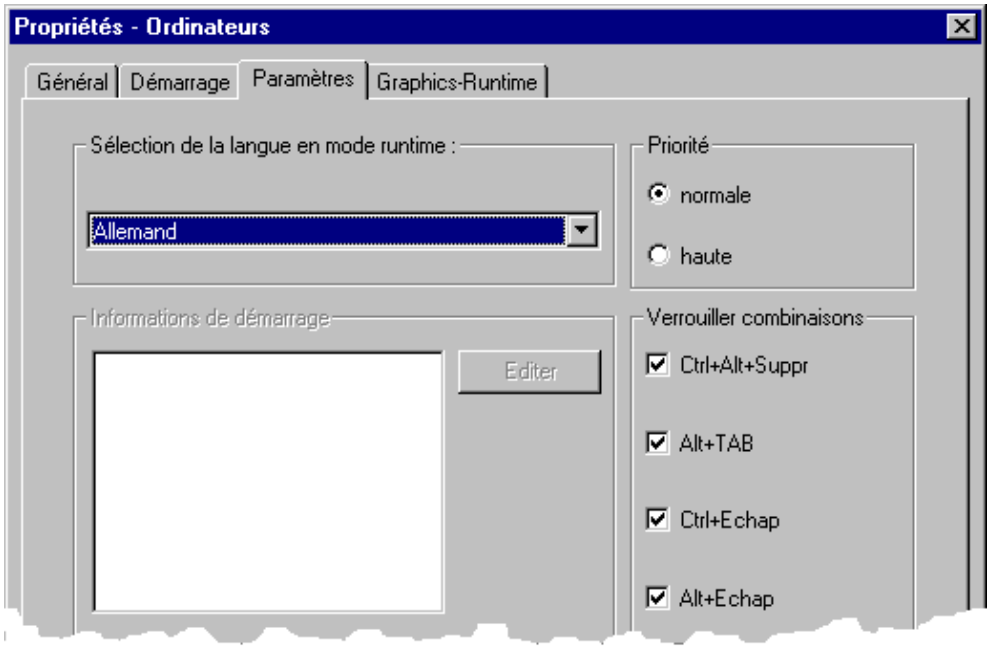
Pas de sélection de:

L'opérateur ne doit pas avoir la possibilité de sélectionner les modules ci-dessous en runtime:

- *Control Center* de WinCC (environnement de configuration),
- la fenêtre de runtime de la *base de données SQL* de WinCC (Sybase SQL Anywhere), un utilisateur pouvant sinon terminer la connexion à la base de données. WinCC n'est alors plus exécutable,
- la *barre de tâches* de Windows, celle-ci pouvant démarrer **tous** les programmes installés,
- la fenêtre de la tâche courante, l'application pouvant être quittée.

Paramétrage nécessaire de l'ordinateur

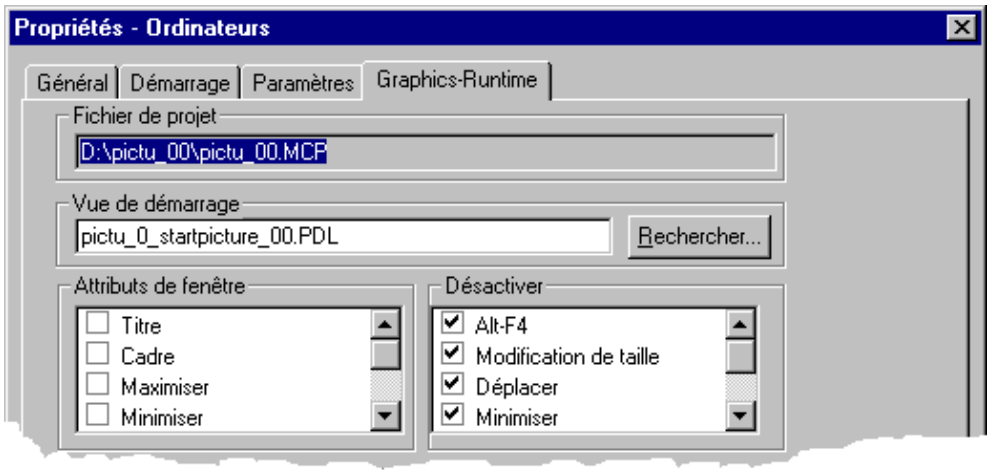
Pour interdire ces manipulations à l'opérateur, les combinaisons de touches ci-dessous doivent être verrouillées.



Les combinaisons de touches se verrouillent dans *Control Center* sous *Propriétés - Ordinateurs*. La signification exacte des diverses combinaisons de touches est indiquée dans l'aide de WinCC et dans l'aide du système d'exploitation correspondant.

Paramétrage nécessaire du runtime

Les touches standard de Windows pourraient fermer la vue de process ; ceci pourrait faire quitter WinCC. Pour éviter ceci, les propriétés de la vue de process doivent être paramétrées de la manière suivante :



Les combinaisons de touches se verrouillent dans *Control Center* sous *Propriétés - Ordinateurs*. La signification exacte des diverses combinaisons de touches est indiquée dans l'aide de WinCC et dans l'aide du système d'exploitation correspondant.

- L'interface utilisateur du système d'exploitation est à nouveau accessible lorsque les commandes *Modification de taille* et *Minimiser* ne sont pas désactivées.
- *Fermer* (pas représenté dans la vue) doit également être désactivé. Sinon le runtime peut être quitté et l'utilisateur bascule dans le système de configuration CS.

Nota:
Lorsque les touches mentionnées sont verrouillées en totalité ou en partie, la possibilité doit être donnée au technicien chargé de la configuration ou au personnel de maintenance d'accéder à la configuration au moyen d'une touche spécialement configurée à cet effet. Ceci s'applique également à l'arrêt ordonné du système informatique.

Ces fonctions ne doivent pas pouvoir être utilisées librement. Une **protection d'accès**, p. ex. pour le personnel maintenance, doit être mémorisée derrière la propriété *Mot de passe* du bouton.

3.3.6 Arrêt ordonné de WinCC

Arrêt par commande opérateur

Une station WinCC ne doit jamais être mise hors tension sans que le système d'exploitation ait été préalablement arrêté. Le système IHM n'est pas adapté à l'utilisation d'un *interrupteur d'arrêt urgence*. Il devra donc être configuré à l'intention de l'opérateur une touche de commande correspondante lui permettant de quitter le système de manière ordonnée sans disposer de connaissances supplémentaires.

Coupures de courant

Un **concept de sauvegarde de données** détaillé doit être élaboré et appliqué pour le système IHM afin d'éviter dans la mesure du possible des pertes de données par variation de tension ou coupure du courant.

Prévoir impérativement pour la station WinCC une alimentation électrique ininterruptible. Celle-ci pourra être réalisée par connexion à un onduleur interne ou par connexion d'un onduleur dédié au serveur WinCC. Ceci est valable aussi bien pour les systèmes monoposte que multiposte, indépendamment du système d'exploitation utilisé (Windows 95 ou Windows NT).
L'onduleur utilisé doit en outre comporter un logiciel spécifique de gestion de coupures de tension pour Windows 95 ou Windows NT, assurant après écoulement d'un laps de temps, l'arrêt sans perte de données du système d'exploitation et de toutes les applications actives en cas de coupure et après mise hors tension du système ; p. ex. : APC USV 600 avec logiciel de gestion de coupure pour Windows 95 et Windows NT.

Remarques sur l'installation d'un onduleur

Le branchement d'un onduleur avec logiciel associé suppose la disponibilité d'une interface série. Si aucune interface série n'existe sur la station WinCC, parce qu'elle est p. ex occupée par une imprimante ou un API, il est nécessaire d'utiliser une carte d'interfaçage supplémentaire.
Les interfaces série à affectation multiple (p. ex. par switch) ne sont pas supportées par la plupart des systèmes d'onduleur et ne sont pas souhaitables non plus car une surveillance permanente du système est alors nécessaire.
Un service de surveillance est installé dans le système d'exploitation. Ce service de surveillance doit posséder des paramètres d'arrêt afin de garantir l'arrêt ordonné du système. Le processus d'arrêt du logiciel applicatif doit être dans tous les cas activé pour que WinCC puisse être arrêté sans perte de données en cas de coupure. Le *temps de sauvegarde* avant coupure sélectionné doit être suffisamment long et tenir compte des applications actives.

Le logiciel d'onduleur possède généralement aussi un *arrêt temporisé*, p. ex. pour le week-end ou la nuit. Ce dispositif permet de **quitter le système WinCC sans manipulation**.

3.3.7 Sauvegarde des données

Quand doit-on sauvegarder?

Un projet WinCC doit être sauvegardé à plusieurs reprises pour des raisons suivantes:

- Sauvegarde des données pendant la phase de configuration.
- Sauvegarde avant exportation et importation de données (p. ex. à l'importation de variables, de textes de vue multilingues, de textes d'alarmes et de textes d'alarmes multilingues).
- Sauvegarde avant la reconstitution ou le déchargement de la base de données WinCC.
- Sauvegarde avant le traitement de la base de données avec des outils tels que Accès SQL interactif.
- Sauvegarde des données de configuration pour l'installation sur l'ordinateur cible du client final.
- Reprise de données pour un projet de structure similaire.

Quels sont les supports de mémorisation appropriés?

Support	Avantage	Inconvénient
Disquettes	lisibles presque partout	trop faible capacité (même avec compactage)
Disquettes ZIP	bon marché, capacité suffisante, accès direct et rapide par Windows ; facilité d'installation ; mobile, pour l'utilisation sur le système informatique	
Lecteur de bande (p. ex. sur réseau)	possibilité de sauvegarde automatique (journalière) ; très grande capacité	disponible généralement qu'en environnement bureautique, pas d'accès direct aux données car utilisant un format spécial pour la mémorisation
Disque dur sur autre PC (p. ex. Laplink)	pas de manipulation de supports physiques ; données directement utilisables	lent ;pas approprié pour les gros volumes de données
Disque MO	grande sécurité des données ; réutilisabilité ; possibilité de sauvegarde des alarmes et valeurs de mesure au runtime.	un lecteur spécial est nécessaire pour l'écriture et la lecture
CD-ROM	capacité élevée ;lisible presque partout ; adapté à l'archivage longue durée.	un lecteur spécial est nécessaire pour la gravure ; support pas réutilisable

Epurement des projets avant la sauvegarde des données

Afin de sauvegarder un projet sous une forme aussi **épurée** que possible pour remise au client final ou pour la réutilisation de données de projet, les données ci-dessous peuvent être supprimées ou épurées à l'aide de programmes additionnels.

- Tous les fichiers de sauvegarde du répertoire *\GraCS*.sav* du projet.
- Si l'utilisateur n'a pas créé ses propres modèles (Report Designer) pour la documentation, les modèles système peuvent être supprimés dans le répertoire *\Prt*. Lors de la création d'un nouveau projet, tous les modèles système sont automatiquement copiés du répertoire *\Siemens\WinCC\syslay* dans le répertoire de projet.

Que doit-on sauvegarder?

Lorsque seules les données d'un projet WinCC doivent être sauvegardées, effectuez la sauvegarde des fichiers et répertoires ainsi que de leurs fichiers suivants.

A partir du répertoire de projet:

- les fichiers **.mcp, *.pin, *.db*
- les répertoires *\GraCS* et *Library*
- le répertoire *\Pas si l'utilisateur a créé ses propres actions*
- le répertoire *\Prt* si l'utilisateur a créé ses propres mises en page d'impression

Lorsque des composants communs à tous les projets (fonctions standard, objets de la bibliothèque de projets) ont été créés, les fichiers ci-après

- les fichiers *\Siemens\WinCC\aplib*.fct*
- le fichier *\Siemens\WinCC\aplib\library.pxl*

doivent être sauvegardés à partir du répertoire standard WinCC. Ces données **ne sont pas** générées lors d'une nouvelle installation de WinCC.

3.3.8 Copie d'un projet WinCC sauvegardé sur une machine cible

Installation du logiciel système

Installation du logiciel WinCC. Sur les systèmes complets, ceci s'effectue à l'aide du dialogue de configuration appelé automatiquement ou avec le CD d'installation livré avec WinCC et les disquettes de licence associées.

Logiciels additionnels

Si votre projet utilise des logiciels additionnels spéciaux (p. ex. packages optionnels ou Add'ons) ou des interfaces de communication spéciales ou encore des interfaces à d'autres programmes Windows (p. ex. WORD, EXCEL, etc.), ces logiciels doivent être également installés sur l'ordinateur cible. Les autorisations pour logiciels optionnels, Add'ons ou interfaces de communication (DLL de canaux utilisés) doivent également être installés sur la machine cible. Notez que **toutes** les autorisations nécessaires (pour tous les DLL de canaux) doivent être chargées avant de pouvoir travailler avec le projet WinCC.

Logiciel Windows

Lorsque les vues WinCC utilisent des liaisons OLEà d'autres programmes Windows, p. ex. WORD, ClipArts ou EXCEL, le logiciel correspondant doit être également installé sur la machine cible en fonction du type de liaison OLE, c'est-à-dire qu'il doit être enregistré dans la base de registres de Windows.

OCX, ActiveX

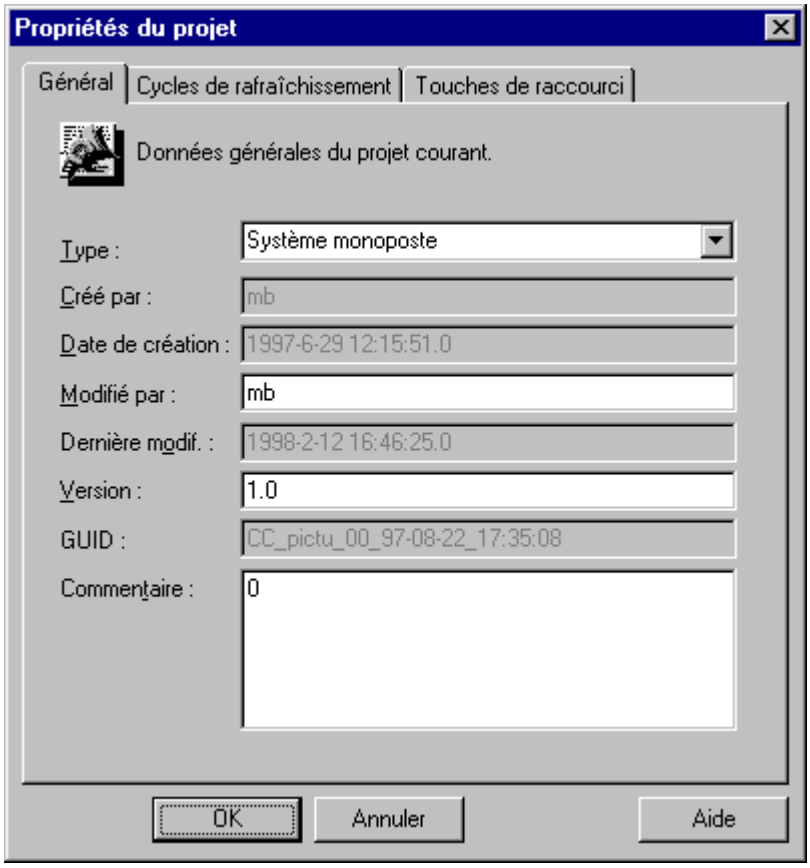
Lorsque d'autres composants OCX (Contrôles OLE, ActiveX) provenant de packages complémentaires sont utilisés, ceux-ci doivent également être enregistrés dans la base de registres de Windows. Enregistrez et vérifiez les enregistrements de ces composants OCX dans la base de registres, p. ex. à l'aide de l'outil livré avec WinCC *SmartTools\CC_OCX_REG\ocxreg.exe*. Si le système ne trouve pas un ennment d'objets OLE ou d'objets OCX au lancement du projet WinCC en runtime (ainsi que dans la configuration avec *Graphics Designer*), cet objet est affiché avec le message *Objet inconnu*.

Réseau

Si le projet a été configuré pour un système **multiposte**, l'installation complète du réseau doit avoir été effectuée sur la machine cible WinCC avant de copier les données WinCC. Notez les noms d'ordinateur nécessaires dans l'environnement d'ordinateurs configuré, ceux-ci étant demandés pour le paramétrage du projet copié. Le nom d'ordinateur est également nécessaire pour paramétrer un système monoposte ; c'est pourquoi vous devez connaître dans tous les cas les noms attribués sur la machine cible ou les déterminer à l'aide du panneau de configuration Windows. S'il s'agit d'un système monoposte sans connexion réseau, le nom d'ordinateur Windows sera entré.

Copie des données et démarrage du projet

Etape	Procédure pour la réutilisation de données
1	Création d'un répertoire de projet (p.ex. *Projets_WinCC*)
2	Copiez l'ensemble du chemin sauvegardé dans le répertoire ainsi créé comme sous-répertoire (p. ex. :*\WinCC_Projets\Varia_00*). Le nom du répertoire de projets WinCC peut être modifié si on le souhaite.

Etape	Procédure pour la réutilisation de données
	Lorsque les fichiers sont renommés dans le répertoire de projets (varia_00.mcp, varia_00.db, varia_00.pin, varia_00.log), veillez à ce que tous les fichiers aient bien le même nom (sauf l'extension).
3	Ouvrir le projet dans WinCC *Control Center* .
4	Modifiez le paramétrage spécifique au projet si nécessaire. Pour la modification du type, des adaptations supplémentaires sont nécessaires dans les *Propriétés - Ordinateurs*. Ces adaptations sont décrites dans l'aide WinCC. 
5	Vérifiez le *nom de l'ordinateur* et le modifier le cas échéant dans l'onglet *Général* du dialogue *Propriétés - Ordinateurs*. Si le nom d'ordinateur du projet WinCC ne coïncide pas avec le nom d'ordinateur du système cible, un message d'erreur est affiché à l'activation du projet, c.-à-d. au démarrage du runtime .
6	Si l'on a utilisé dans le projet ses propres *fonctions standard* (*.fct), copiez les fonctions standard sauvegardées dans le chemin standard WinCC *\Siemens\WinCC\aplib* et déclarez-les ensuite dans l'arbre des fonctions. • Appelez l'éditeur *Global Script* dans le projet ouvert • Recréez la structure de déclaration avec *Options → Générer en-tête* . Les nouvelles fonctions sont alors visibles dans l'arbre des fonctions.
7	Activez le projet pour vérifier le lancement correct du projet.

3.3.9 Réutilisation - Reprise d'éléments de projet dans un nouveau projet ou dans un projet existant

Raison de la réutilisation

Un projet WinCC peut naître de différentes façons. Les aspects les plus importants sont la réutilisabilité d'éléments existant déjà dans des projets semblables ou la reprise de données dans les exemples de projet.

Equipe de projet

En cas de programmation en équipe, les tâches à réaliser sont similaires car en fin de compte les éléments du projet WinCC devront être fusionnés en un seul projet.
Un projet WinCC est constitué de fichiers (p. ex. vues) **et** des données de configuration stockées dans la base de données (système d'alarmes, gestion des variables).

Données stockées dans la base de données

Les données rangées dans la base des données de configuration **ne peuvent pas** être créées dans deux projets individuels et ensuite regroupées.
C'est pourquoi il faut, lors de la configuration d'informations de base de données (p. ex. conception des systèmes d'alarmes), créer un **projet type** qui sera utilisé pour ce genre de configuration. Ce projet type doit être **sauvegardé** avant **chaque** modification du contenu de la base de données (également lors d'opérations intermédiaires). En cas d'incident pendant la modification des données, l'état des données antérieur à la modification est alors disponible.

Nota:
Notez que toute modification du contenu de la base de données influe sur la structure et l'accès à la base de données. Un grand nombre de modifications superflues (avec le cas échéant des suppressions de données) peuvent avoir des conséquences néfastes sur l'organisation de la base de données WinCC (perte de performances).

Système monoposte

Pendant qu'est exécutée l'opération de configuration suivante, p. ex. du système d'alarmes, dans le projet type, ne modifiez en aucun cas la base de données en un autre endroit sur un système WinCC *monoposte* (p. ex. archivage – *Tag Logging*).

Système multiposte

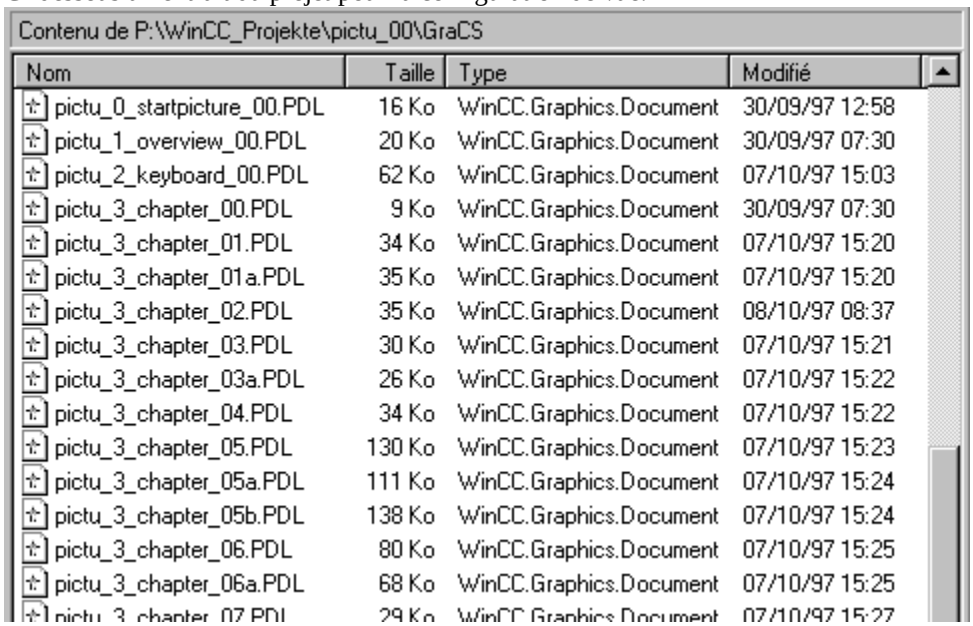
Si, par contre, un projet est configuré sur un système WinCC *multiposte*, la configuration peut avoir lieu simultanément dans d'autres domaines de la base de données. Une personne peut p. ex. configurer le système d'alarmes et une autre le système d'archivage (acquisition de valeurs de mesure par exemple).

Conversion système monoposte - multiposte

Tous les projets peuvent être créés sur un système de configuration WinCC *multiposte* et être ensuite convertis en un projet pour système monoposte sur la machine cible. Lorsque l'on prévoit le passage de station(s) de configuration en système client, n'utiliser aucun élément WinCC spécifique conçu pour systèmes multiposte. Ne pas utiliser, par exemple, de variables internes locales, spécifiques à un ordinateur, si le système doit être installé en monoposte chez le client.
On doit savoir à la création d'un nouveau projet (monoposte ou multiposte) si un projet type existant doit être utilisé comme point de départ, avec son système d'alarmes préconfiguré et ses données d'archivage. Les données rangées dans la base de données ne sont transférables dans d'autres projets que par reconfiguration ou, lorsque cela est possible, par exportation - importation.

3.3.9.1 Réutilisation de vues

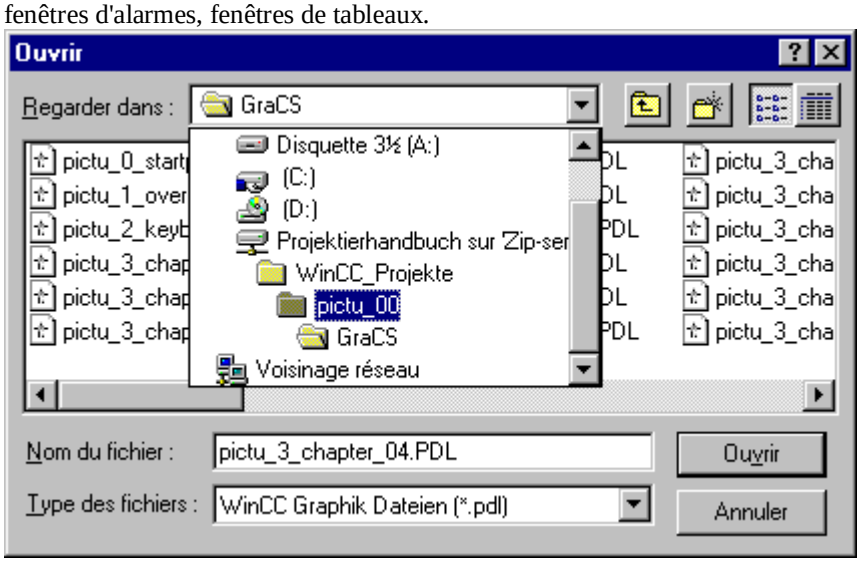
Les vues configurées peuvent être réutilisées à tout moment. Vous pouvez procéder de deux manières : copier les fichiers de vue (**.pdl*) directement, dans l'explorateur Windows depuis le répertoire source dans le répertoire cible accessible par le chemin de projet WinCC *\GraCS* (méthode appropriée pour plusieurs fichiers de vue);
Ci-dessous un extrait du projet pour la configuration de vue.



La deuxième méthode de réutilisation de vues consiste à ouvrir un fichier de vue (*vue.pdl*) dans *Graphics Designer* avec la commande *Fichier → Ouvrir fichier.* La vue est ensuite enregistrée dans le répertoire courant des vues (*\GraCS*) à l'aide de la commande de menu *Fichier → Enregistrer sous* . Cette procédure est bien adaptée lorsque les fichiers de vue doivent être utilisés comme point de départ et adaptés immédiatement.

Restrictions

La réutilisation fonctionne également, quoiqu'avec certaines restrictions, pour les cadres de courbes, fenêtres d'alarmes, fenêtres de tableaux.



Nota:
Les références d'objets réutilisées pour des objets dans le projet existant ne correspondent plus. Il est nécessaire de procéder à une redéfinition des références correspondantes dans le nouveau projet.

Ces références peuvent être:

Références dans les vues

- Structures du domaine Type de données
- Variables internes ou externes
- Variables système
- Modèles de fenêtres d'alarmes
- Modèle de fenêtre d'archive (fenêtre de courbes ou de tableaux pour archive de valeurs de process ou archive utilisateur)
- Objets de vue mémorisés au format Bitmap ou comme métafichier (p. ex. pour les indicateurs d'état ou les objets graphiques)
- Autres vues utilisées comme boîtes de graphisme ou de process ou fenêtres affichées
- Fonctions de projet utilisées
- Droits d'accès

Les références suivantes doivent en outre être définies :

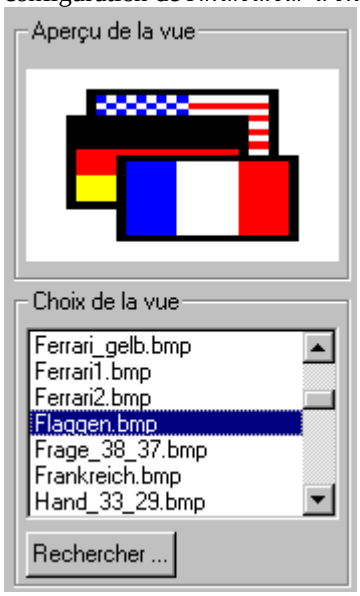
- Définition des structures associées Types de données, p. ex. structures de régulateurs ou de modèles pour objets utilisateur
- Définition des canaux de communication et des liaisons logiques avec définition des variables (éventuellement avec groupes de variables)
- Définition des variables internes et noms des variables système (commençant par @)
- Définition de modèles de fenêtre nouveaux ou semblables et liaison des fenêtres d'application (pour *Alarm Logging* et *Tag Logging*)
- Réutilisation des éléments de vue (*.bmp ou *.emf) par copie à partir du répertoire *\GraCS*

- Réutilisation des contenus des fenêtres de vue par copie des autres fichiers de vues (*.pdl) depuis le répertoire *\GraCS*
- Les fonctions de projet utilisées doivent être copiées dans le répertoire *\library* du nouveau projet depuis le projet source. De plus, ces fonctions doivent être rangées dans l'arbre de fonctions avec l'éditeur *Global Script* via *Générer en-tête*. Cette procédure a déjà été décrite de manière plus détaillée dans le chapitre *4.1 Environnement de développement de scripts dans WinCC*
- Définition des droits d'accès utilisés dans l'éditeur *User Administrator*. Les droits d'accès utilisés (p. ex. pour les touches de commande) doivent être attribués pour les définitions de groupes.

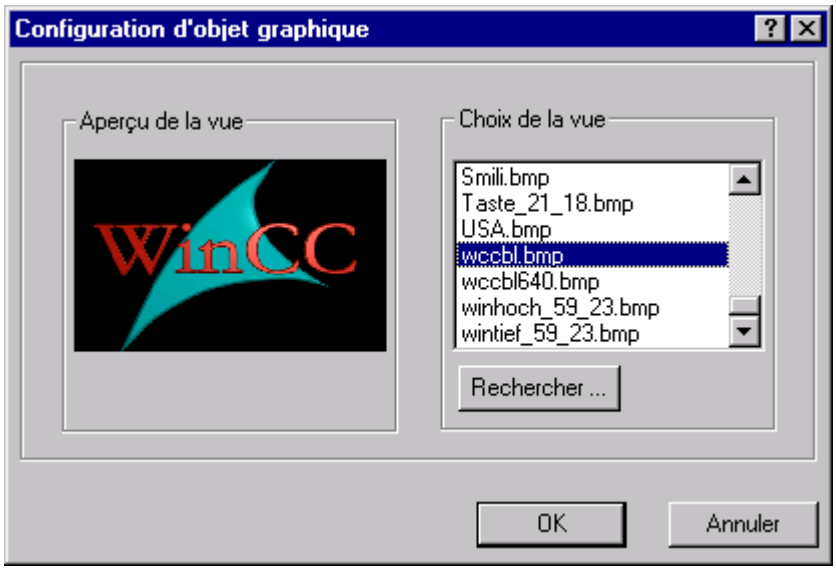
3.3.9.2 Réutilisation de symboles et de bitmaps

Copie

Les symboles utilisés pour *l'affichage des états* et les *objets graphiques* des fichiers de vue sont mémorisés sous forme de fichiers propres dans le répertoire de vues du projet. Pour cela, copiez les fichiers de symboles souhaités (*.emf ou *.bmp) dans le répertoire cible *\GraCS* du nouveau projet. Ces vues sont alors immédiatement disponibles dans la liste visualisée pour les *indicateurs d'état* ou les *objets graphiques* (voir palette d'objets dans *Graphics Designer*). Extrait du dialogue de configuration de l'*indicateur d'état*.



Sélection de vue pour l'*objet graphique*:



Importation

Les symboles peuvent être liés à une vue soit par la méthode décrite ci-dessus soit directement, par *Ajouter* → *Importation* d'un symbole dans la vue graphique en cours de traitement. Il n'y a pas dans ce cas de fichier à copier : vous importez le symbole souhaité directement en accédant au chemin du projet source (*GraCS*) et au fichier de symboles souhaité (*.bmp, *.emf, *.wmf). Après l'importation, le symbole apparaît directement comme objet dans la vue (en haut à gauche).

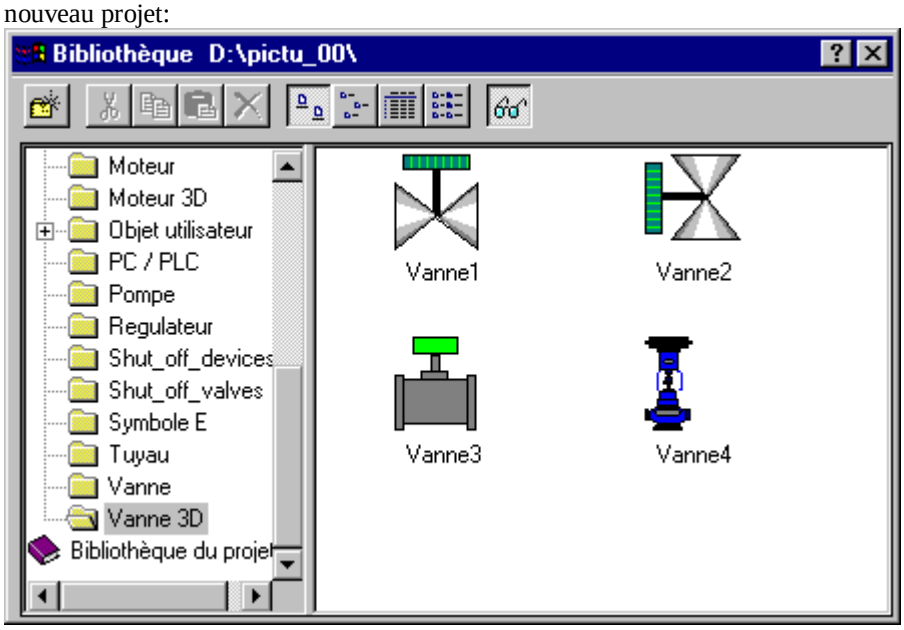
Lorsque des symboles ont été rangés dans une bibliothèque de projets, cette bibliothèque de projets peut être réutilisée dans un autre projet par reprise de la totalité de la bibliothèque de projets. Voir description ci-dessous.

3.3.9.3 Réutilisation d'une bibliothèque de projets (contenant des symboles et des objets utilisateur préconfigurés)

Bibliothèque globale

Lorsque des symboles ont été rangés dans une bibliothèque de projets, celle-ci peut être réutilisée dans un autre projet par copie du fichier *library.pxl* dans le chemin *\library*.

Les composants préconfigurés peuvent être maintenant réutilisés à n'importe quel moment dans le nouveau projet:

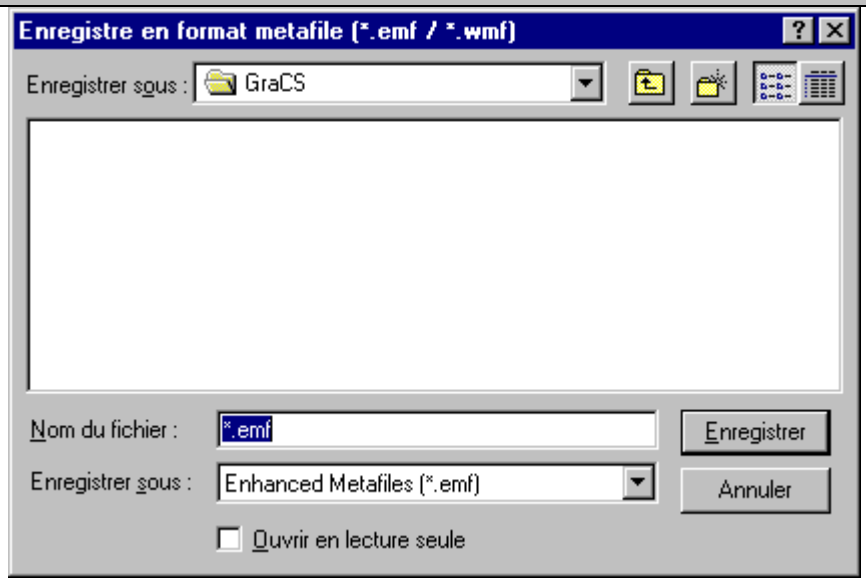


Nota:
Notez que les symboles liés peuvent renvoyer à des références (ou des variables inexistantes) que vous devez donc tout d'abord définir. Les actions ou liaisons correspondantes doivent être éventuellement adaptées en fonction de la configuration de ces symboles. Il faut donc vérifier après utilisation de symboles repris dans la bibliothèque quelles propriétés/liaisons à des événements existent déjà ou si ces propriétés/liaisons doivent être éventuellement adaptées.

Symboles individuels

Lorsque certains symboles de la bibliothèque de projets sont réutilisés dans un nouveau projet, ceux-ci sont exportés individuellement (fichier symbole *.emf).

Etape	Procédure: Réutilisation de symboles
1	Ouvrir bibliothèque
2	Sélectionnez le symbole souhaité avec et amenez le symbole dans la vue par glisser-déposer en maintenant enfoncé le bouton de la souris.
3	Sélectionnez *Fichier →Exportation...* pour appeler le dialogue d'enregistrement du symbole.

Etape	Procédure: Réutilisation de symboles
	
4	Enregistrer le symbole

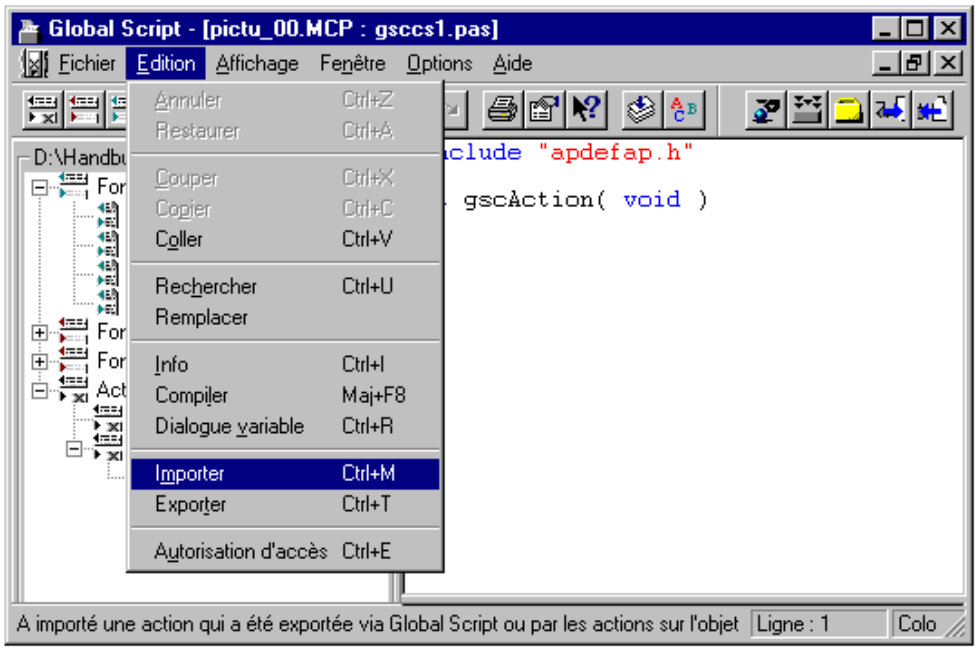
Nouvelle bibliothèque de projets

Ces symboles exportés sont maintenant disponibles sous forme de fichiers de symbole individuels réutilisables individuellement par importation. Si ces symboles sont utilisés fréquemment dans le projet, ils doivent être ajoutés à la nouvelle bibliothèque de projets. Pour cela, appelez la bibliothèque de symboles et particulièrement la bibliothèque de projets :

Créez votre propre répertoire de symboles, p. ex. à l'aide du symbole de répertoire de la barre d'outils de la fenêtre de bibliothèque et copiez par *glisser-déposer* les symboles importés dans ce répertoire. De cette manière, vous pouvez reprendre en partie vos symboles dans des projets existants et ajouter de nouveaux symboles spécifiques afin de recréer une bibliothèque de symboles propre au nouveau projet.

3.3.9.4 Réutilisation d'actions

Les actions utilisées fréquemment dans un projet ou devant être copiées d'une action programmée sur un objet dans une autre action sur un objet sont mémorisées sous forme de fichiers spécifiques. Ces fichiers sont enregistrés dans le répertoire *\GraCS* avec l'extension *.act*. Ils peuvent être repris à tout moment par copie depuis le répertoire source dans le répertoire cible.
L'enregistrement d'un fichier d'action s'effectue via l'*éditeur d'action C*, au moyen du bouton *Exporter action* de la barre d'outils, dans le fichier cible que vous nommez vous-même (avec l'extension *act* signifiant action).



Le bouton *Importer action* permet de reprendre un fichier d'action mémorisé pour une action sur un objet dans une vue du nouveau projet. Lire attentivement à ce sujet le chapitre *4.1 Environnement de développement de scripts dans WinCC*.

Nota:
Les actions utilisées fréquemment peuvent aussi être définies comme fonctions de projet ou fonctions standard.

3.3.9.5 Réutilisation de variables

La gestion des variables de WinCC peut être complétée de différentes manières :

- Ecriture de variables de données S5 ou de variables de données S7 à l'aide de l'assistant (*Dynamic Wizard*)
- Réutilisation de *variables S7* au moyen du *Mapper PCS7*
- Importation et exportation de listes de textes à l'aide de l'outil Add-on *Var_Exim*
- Accès conversationnel aux tables de la base de données (tables de variables)
- Assistants spécifiquement programmés à cet effet (*Dynamic Wizard*) ou programmes qui créent de nouvelles données dans la gestion de variable à l'aide des fonctions API de WinCC

Les deux dernières possibilités supposent de très bonnes connaissances du maniement des bases de données SQL et de la programmation par interface de programmation d'application (API). Ceci ne doit être effectué que par des personnes possédant de telles connaissances.

Avant de reprendre les données dans le projet cible il faut par conséquent examiner où se trouve la base du projet cible. S'il existe déjà un grand nombre de variables dans la gestion des variables de WinCC, il convient d'importer la liste de variables de WinCC dans le projet cible. Les *variables internes* doivent toujours être reprises de la *Gestion des variables* de WinCC. Utilisez pour cela l'outil *Var_Exim.exe*.

Réutilisation de variables de données S5-/S7 avec assistant Dynamic Wizard

Les définitions de domaines de données créées avec le logiciel STEP5/STEP7 peuvent être écrites dans la gestion des variables de WinCC à l'aide des assistants *Dynamic Wizards* existants. Effectuez les opérations suivantes.

Etape	Procédure: Réutilisation de données S5 ou S7
1	Effectuez une **sauvegarde des données** du projet. Des modifications sont réalisées dans la base de données.
2	Exportez la liste de correspondance avec le logiciel STEP. Un fichier *prj_zuli.SEQest créé.*
3	Eliminez de ce fichier exporté les symboles spéciaux (p. ex. pour appels de programme) qui ne sont pas nécessaires pour l'importation dans WinCC. Ceci peut s'effectuer avec un éditeur de texte (p. ex. Wordpad). La liste de correspondance ne doit pas contenir de lignes vides.
4	Ouvrez le projet cible *dans* WinCC *Control Center*. Le projet doit être en mode configuration (runtime pas actif).
5	Ouvrez l'éditeur *Graphics Designer*. Sélectionnez dans une vue quelconque de l'assistant *Dynamic Wizard* (par les commandes *Affichage → Barre d'outils...*) l'onglet *fonctions d'importation*. Sélectionnez la fonction *Importation S7 – S5 ZULI* . Indiquez ensuite le nom du *fichier source (.seq)* en mentionnant le répertoire (à l'aide du bouton). Spécifiez en outre la liaison logique par laquelle les descriptions de variable seront rattachées à la liste d'affectations. Les données sont alors entrées dans la gestion des variables WinCC. Les noms de variables doivent être uniques dans tout le projet WinCC. Les variables sont ajoutées à la gestion des variables existant déjà le cas échéant. Le nom de variable constitue la clé.

Réutilisation de variables avec programme auxiliaire

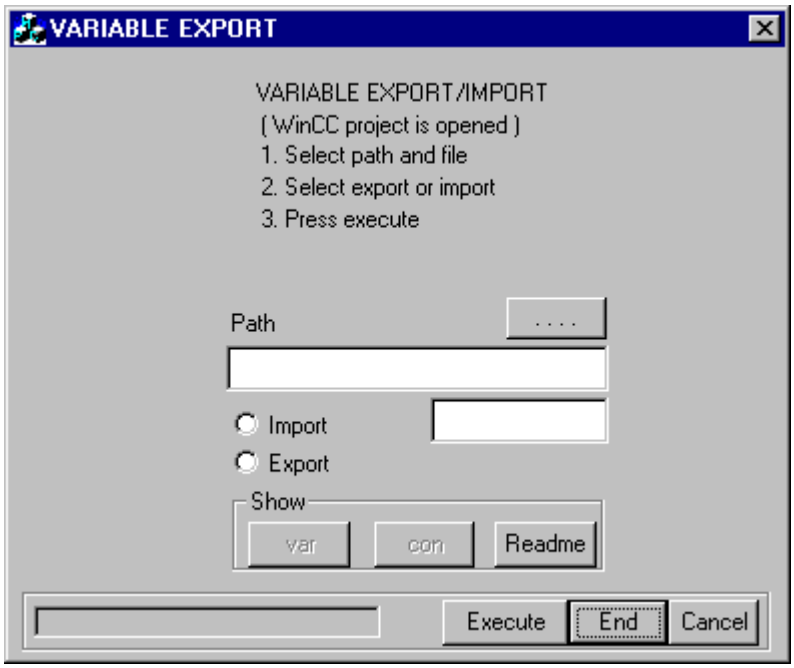
Les liaisons (DLL de canal, liaisons logiques et paramètres de liaison) doivent avoir été déjà définies dans le projet cible avant l'importation.

Nota:
Pour créer automatiquement des liaisons et entrer automatiquement des données dans la base de données WinCC, on pourrait créer un programme spécial réalisant de telles définitions par l'interface de programmation WinCC (API). De cette manière, les données de process existant déjà peuvent être complétées automatiquement. Ceci doit être réalisé par un spécialiste de la programmation par interfaces API WinCC ou de programmation SQL.

Les variables définies dans la gestion des variables peuvent être à tout moment **exportées** comme fichiers texte pour compléter la liste des variables. Les données ainsi créées doivent être **réimportées** dans la gestion des variables du projet. Les fichiers créés sont au format CSV (comma separated version) et peuvent être lus et traités par n'importe quel programme de mise en forme.
Il existe pour cela une application rangée dans le répertoire *\SmartTools\CC_VariablenImportExport* du CD-ROM. Ce programme Windows est fourni comme Add-on:

- Pour exportation des données de la gestion des variables
- Pour l'importation de données de variables déjà créées en externe
- Pour la configuration de données de masse

Les opérations ci-dessous sont nécessaires pour l'exportation ou l'importation de données.

Etape	Procédure: Importation et exportation de variables
1	Ouvrez votre projet WinCC dans WinCC *Control Center*.
2	Définissez les liaisons encore inexistantes mais nécessaires pour l'importation ultérieure (DLL de canal – liaison logique – paramètres de liaison). Ceci ne doit cependant être exécuté que dans le nouveau projet. Une nouvelle opération d'exportation/importation peut être nécessaire.
3	Activer le programme *Var_exim* par **GG**. L'interface utilisateur de ce programme est présentée ci-après. 

Importation - Exportation

Le paramétrage ci-dessous est nécessaire pour l'exportation ou l'importation :

Localisation, action	**Importation**	**Exportation**
Path	Sélectionnez le répertoire du projet contenant les fichiers pour l'importation de variables. La sélection s'effectue en choisissant le fichier <*projektname*>.*mcp*. Les fichiers contenant les données pour importation doivent se trouver dans le même répertoire que le fichier de projet.	Indiquez le répertoire pour l'exportation de variables. La sélection s'effectue en choisissant le fichier <*projektname*>.*mcp*.
Action	Sélectionnez *importation* Si les données de variable existantes doivent être écrasées, sélectionnez *Import overwrite*.	Sélectionnez *exportation*.
EXECUTE	Sélectionnez Exécuter Le dialogue alors affiché montre les paramètres réglés et exécute le transfert après validation avec le bouton *OK*. Les vérifications effectuées font que l'importation demande plus de temps.	Sélectionnez Exécuter Le dialogue alors affiché montre les paramètres réglés et exécute le transfert après validation avec le bouton *OK*.
Indicateur d'état	Fin Exportation/Importation	Fin Exportation/Importation
Fichier de variables *Name_vex.csv*	Base de l'importation : constitué d'une ligne d'en-tête et d'enregistrements	La liste de variables créée est mémorisée sous forme de texte dans ce fichier. Son contenu peut être consulté directement avec le bouton *var* ou édité dans un éditeur de texte (Notepad) ou avec EXCEL.
Fichier de variables *Name_cex.csv*	Base de l'importation : constitué d'une ligne d'en-tête et d'enregistrements (champs de structure)	Ce fichier contient les liaisons courantes configurées auxquelles se réfère le fichier de variables. Son contenu peut être consulté directement avec le bouton *con* ou édité dans un éditeur de texte (Notepad) ou avec EXCEL.
Fichier de structure des données *Name_dex.csv*	Base de l'importation : constitué d'une ligne d'en-tête et d'enregistrements	S'il existe des variables de type structure, un fichier contenant des informations de structure est créé en plus. Son contenu peut être édité avec un éditeur de texte (Notepad) ou avec EXCEL.
Fichier de diagnostic *Diag.txt*	Fichier de diagnostic avec indication des variables n'ayant pas pu être importées.	

Le bouton *End* permet de quitter le programme.

Liste de variables

Le tableau ci-après décrit la structure des listes de variables.

Champ	Type	Description
Varname	char	Nom de variable:
Conn	char	Liaison
Group	char	Nom de groupe
Spec	char	Variable interne ou adresse (cohérente avec le type de liaison)
Flag	DWORD	
Common	DWORD	
Ctyp	DWORD	Type de variable 1 BIT 2 SBYTE 3 BYTE 4 SWORD 5 WORD 6 SDWORD 7 DWORD 8 FLOAT 9 DOUBLE 10 TEXT_8 11 TEXT_16 12 Type données brutes 13 Tableau 14 Structure 15 CHAMP DE BITS_8 16 CHAMP DE BITS_16 17 CHAMP DE BITS_32 18 Référence de texte
CLen	DWORD	Longueur des variables
CPro	DWORD	Variable interne ou externe
CFor	DWORD	Transtypage
Protocol		
P1	BOOL	Violation seuil supérieur
P2	BOOL	Violation seuil inférieur
P3	BOOL	Erreur transtypage
P4	BOOL	Erreur d'écriture
P5	BOOL	
P6	BOOL	
L1	BOOL	Valeur de substitution pour violation seuil supérieur
L2	BOOL	Valeur de substitution pour violation seuil inférieur
L3	BOOL	Valeur de démarrage
L4	BOOL	Valeur de substitution pour erreur de liaison
L5	BOOL	Seuil supérieur valide
L6	BOOL	Seuil inférieur valide
L7	BOOL	Valeur de démarrage valide
L8	BOOL	Valeur de substitution valide
LF1	double	Seuil supérieur

Champ	Type	Description
LF2	double	Seuil inférieur
LF3	double	Valeur de démarrage
LF4	double	Valeur de substitution
Echelle		
SCF	DWORD	1 si échelle définie
SPU	double	Plage de valeurs process de
SPO	double	Plage de valeurs process à
SVU	double	Plage de valeurs variable de
SVO	double	Plage de valeurs variable à

Liste de liaisons

Champ	Type	Description
Conname	char	Nom de liaison logique
Unit	char	Unité de canal
Common	char	Général
Specifique	char	Paramètres de liaison spécifiques
Flag	DWORD	

Liste de structures de données

Champ	Type	Description
Datastructure	short	Nom de structure ou nom de champ
Type ID	short	Identification (utilisée pour Ctyp dans liste de variables)
Creator ID	short	

Pour pouvoir traiter sous EXCEL (Version 7.0 ou 8.0) les listes de textes, vous devez ouvrir les fichiers exportés avec le type de fichier *fichiers texte [*.prn; *.txt; *.csv]*.

Consignes

Respectez les consignes ci-dessous pour l'adaptation des données de variables dans la liste de textes:

Type	dans liste de textes	dans WinCC
Liaisons	Description générale	Les liaisons logiques **doivent être** définies si elles n'existent pas !
	Description spécifique DLL de canal	Les DLL de canal **doivent être** définies si elles n'existent pas! !
Groupe de variables	Pas d'information de groupe Si des groupes ne contenant aucune variable sont définis dans le projet, ceux-ci ne sont pas exportés.	L'information de groupe est générée automatiquement à la première variable lors de l'importation.
Variables	Description générale	
	Description spécifique DLL de canal ou variable	DLL de canal ou variable interne

Type	dans liste de textes	dans WinCC
	interne correspondantes	
	Les éléments manquants sont remplacés par * à l'exportation.	Les variables ne possédant pas de description spécifique ne sont pas importées!
Variables du type structure	Affectation selon définition de structure de la liste de structures	Affectation au type de données.
Définition structure		Définition par le technicien de configuration.
Valeurs/seuils	Ne sont pas exportés ni importés avec la liste de texte	Définition par le technicien de configuration.

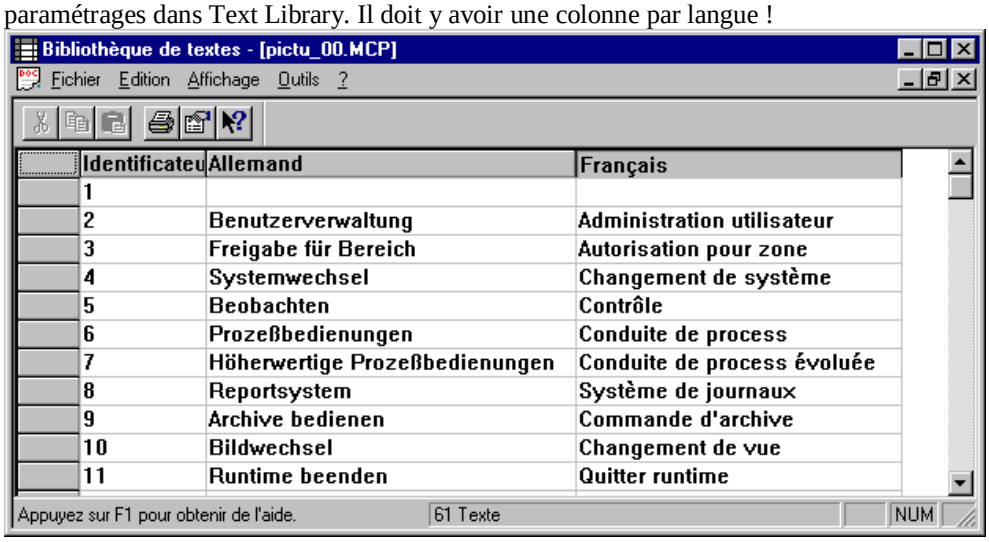
Procédez à une **sauvegarde des données** du projet **avant** d'importer les variables modifiées ou nouvelles. Des modifications sont réalisées dans la base de données. Ces modifications ne peuvent pas être annulées dans WinCC.

3.3.9.6 Réutilisation de textes multilingues (dans vues, alarmes)

Textes multilingues des vues

Les textes multilingues mémorisés dans les vues peuvent être repris dans le nouveau projet par copie de la vue.

Dans le nouveau projet, la langue doit être ajoutée à la bibliothèque de textes. Vérifiez à cet effet les paramétrages dans Text Library. Il doit y avoir une colonne par langue !



Les langues ne doivent-elles être reprises que partiellement dans le nouveau projet ?
La totalité des informations de texte étant mémorisée pour chaque vue, une reconfiguration est nécessaire dans la langue souhaitée pour chaque vue. Il convient de réfléchir à l'opportunité de supprimer les textes déjà configurés. Le basculement au runtime n'est possible que par des boutons spécialement configurés et pourrait constituer une limitation pour le projet. Si les textes doivent être malgré tout supprimés pour une langue déjà introduite, il est recommandé d'effectuer l'exportation ou l'importation avec l'outil Add-on *Language*.

Réutilisation de vues avec références de texte

Lorsque les vues réutilisées utilisent des références de texte, les données ci-dessous doivent être également reprises:

- Réutilisation des variables correspondantes (exportation ou nouvelle définition) de la gestion des variables du projet WinCC
- Réutilisation des textes de la bibliothèque de textes
- Les variables de référence de texte doivent posséder des identifiants de texte valides (ID de texte). Vérifiez que les ID de texte correspondent encore bien aux textes.

Réutilisation de textes de la bibliothèque de textes

Si les textes de la bibliothèque de textes ne sont repris qu'en partie, les identifiants de texte doivent être adaptés. La réutilisation de textes de la bibliothèque de textes s'effectue par le mécanisme d'exportation/importation dans l'éditeur *Text Library*.

3.3.9.7 Réutilisation d'alarmes

Le référentiel des alarmes

- a besoin d'être modifié et
- demande un lourd travail de configuration en raison des grandes quantités de données impliquées (données de masse).

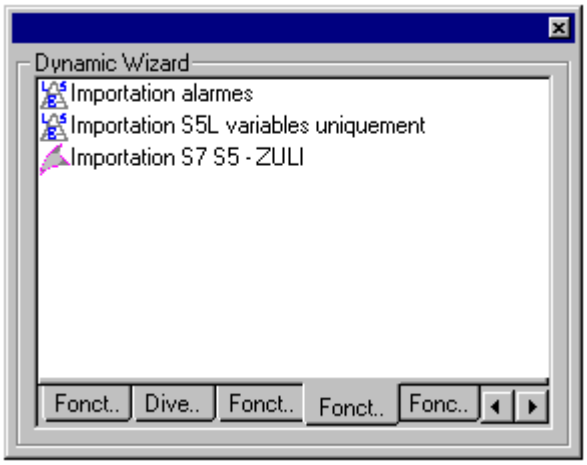
On utilise pour cette raison souvent la réutilisation de données d'alarmes de projets existant déjà. Les modes de réutilisation ci-dessous sont possibles en fonction de la source de données d'alarme :

- Réutilisation des informations d'alarmes déjà configurées dans des systèmes antérieurs (p. ex. COROS)
- Importation d'alarmes (individuelles) d'un projet WinCC existant
- Importation d'informations d'alarmes de la phase de conception

Réutilisation d'alarmes de COROS

La méthode à suivre pour les sources énumérées est la suivante:

Etape	Procédure: Réutilisation de textes d'alarme existant dans COROS
1	Exporter dans COROS les informations d'alarmes (*mldtexte.txt*)
2	Ecrire ce fichier d'alarmes dans WinCC à l'aide de l'assistant *Dynamic Wizard Fonctions d'importation* → *Importation alarmes* . Les données sont alors importées dans le projet WinCC courant.

Etape	Procédure: Réutilisation de textes d'alarme existant dans COROS
	

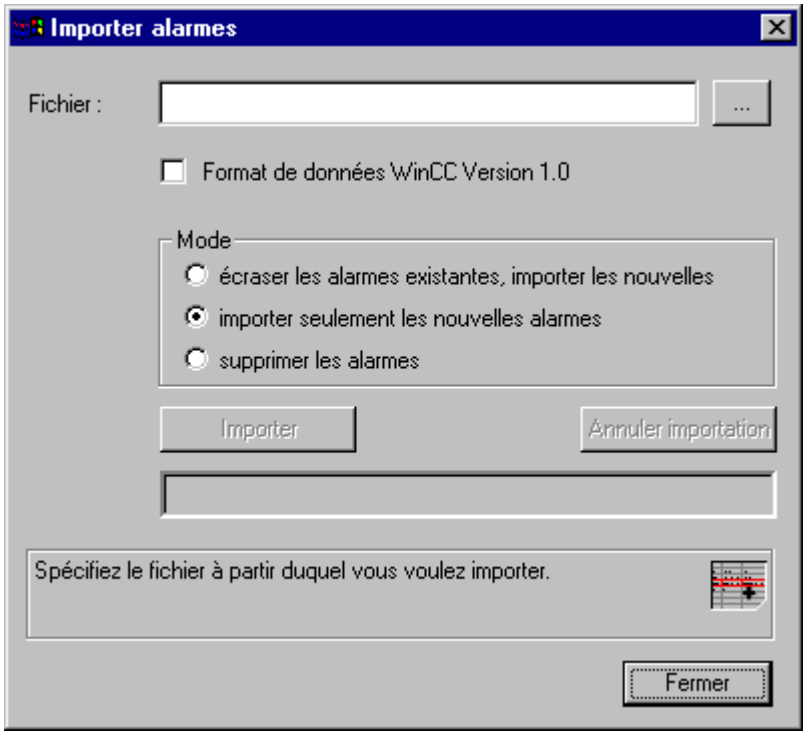
Réutilisation d'alarmes d'un projet WinCC

Lorsque des alarmes sont reprises d'un projet WinCC existant, il est d'abord nécessaire de vérifier si le système d'alarmes cible est construit de manière identique dans la structure de la base de données. Les différences résident par exemple dans les blocs utilisateur et les blocs de valeurs de process. Organisez les blocs de données cible si possible dans le même ordre (et avec la longueur des éléments de texte). Vous devrez sinon procéder à une adaptation dans les différentes colonnes avant l'importation.

Etape	Procédure: Réutilisation de textes d'alarme WinCC existants
1	Ouvrez l'éditeur *Alarm Logging* dans le projet courant.
2	Sélectionnez l'exportation des alarmes par la commande de menu *Alarmes → Exporter alarmes individuelles*.
3	Indiquez le fichier de texte cible dans lequel les informations d'alarmes à exporter doivent être enregistrées. Sélectionnez via *Sélection* les alarmes à exporter à l'aide des critères p. ex. numéro d'alarme, classe d'alarmes.
4	Lancez la procédure d'exportation avec la commande *Exporter*. Un fichier de texte contenant les informations d'alarme, séparées par des virgules, est alors créé.
5	Fermez le projet courant et ouvrez le nouveau projet. Rouvrez l'éditeur *Alarm Logging* et définissez tout d'abord les classes et types d'alarmes nécessaires. Définissez une alarme par type d'alarme afin de disposer de la structure brute pour l'importation qui va suivre.
6	Pour exporter ces alarmes de base, appelez *Alarmes → Exporter alarmes individuelles*. Procédez comme aux points 3 et 4.
7	Ouvrez p. ex. dans EXCEL le fichier d'alarmes du projet source et le fichier d'alarmes du projet cible. Les colonnes sont séparées par des virgules. Comparez la structure des blocs d'alarme et procédez le cas échéant à des adaptations en transposant ou en renommant des colonnes. Entrez dans les blocs contenant les identifiants de textes (ID) l'indice 0. De cette manière, les textes seront automatiquement organisés dans la bibliothèque de textes à l'importation. Les anciens identifiants ne doivent en aucun cas être conservés ! Enregistrez à nouveau le fichier modifié comme fichier de texte
8	Appelez maintenant la fonction importation par les commandes *Alarmes → Importer alarmes individuelles*.

Etape	Procédure: Réutilisation de textes d'alarme WinCC existants
9	Indiquez maintenant le fichier texte source contenant les informations d'alarmes exportées. Vous devez ensuite décider si les alarmes existantes doivent être écrasées à l'importation. L'affectation s'effectue par le numéro d'alarme, qui doit être unique dans le projet.
10	Les alarmes sont ensuite importées et viennent compléter votre système d'alarmes existantes. Contrôlez les affectations importées.

Nota:
Lorsque des données d'alarme sont reprises d'un projet WinCC version 1.10, faire attention aux noms de colonnes dans le fichier d'alarmes !



Importation d'informations d'alarme de la phase de conception avec feuilles de calcul EXCEL

Les informations d'alarme sont disponibles dans une feuille de calcul EXCEL. Ces alarmes peuvent être reprises par transfert des colonnes de la feuille dans la structure d'alarme du projet WinCC. Il faut pour cela partir d'un fichier d'alarmes WinCC, qui se crée de la manière décrite ci-dessous:

Etape	Procédure: Création d'une structure d'alarme
1	Ouvrez le nouveau projet dans WinCC *Control Center*. Ouvrez l'éditeur *Alarm Logging* et définissez tout d'abord les blocs d'alarmes, classes d'alarmes et types d'alarmes nécessaires. Définissez une alarme par type d'alarme afin de disposer de la structure brute pour l'importation qui va suivre.
2	Pour exporter ces alarmes de base, appelez *Alarmes → Exporter alarmes individuelles.*
3	Indiquez le fichier de texte cible dans lequel les informations d'alarmes à exporter doivent être enregistrées.
4	Lancez la procédure d'exportation avec la commande *Exporter*. Un fichier de texte contenant les informations d'alarme, séparées par des virgules, est alors créé.

Etape	Procédure: Création d'une structure d'alarme
5	Ouvrez dans EXCEL le fichier d'alarmes et le fichier d'alarmes nouvellement créé du projet cible. Les colonnes sont séparées par des virgules. Créez une ligne copiée pour la classe d'alarmes/le type d'alarme correspondant dans la feuille de calcul. Reprenez les textes d'alarmes, etc. dans les données source et entrez-les dans les blocs correspondants. Par exemple: *Bloc 1 → Texte d'alarme*. **Numérotez** toutes les lignes d'alarme (p. ex. à partir de 1). Dans EXCEL, ceci peut être effectué rapidement à l'aide de la numérotation dans la colonne des numéros d'alarme.
6	Entrez dans les blocs contenant les identifiants de textes (ID) l'indice 0. De cette manière, les textes seront automatiquement organisés dans la bibliothèque de textes à l'importation. Les anciens identifiants ne doivent en aucun cas être conservés ! Enregistrez à nouveau le fichier modifié comme fichier de texte
7	Appelez maintenant dans l'éditeur Alarm Logging la fonction importation via *Alarmes → Importer alarmes individuelles*.
8	Indiquez maintenant le fichier texte source contenant les informations d'alarmes exportées. Définissez ensuite les paramètres pour que les alarmes existantes soient écrasées. L'affectation s'effectue par le numéro d'alarme, qui doit être unique dans le projet.
9	Les alarmes sont ensuite importées et viennent compléter votre système d'alarmes existantes avec les informations d'alarmes déjà configurées. Contrôlez les affectations importées.

3.3.9.8 Réutilisation de valeurs de mesure

Les définitions des points de mesure, des archives de valeurs de process et des archives utilisateur étant intégrées, avec leurs propriétés, directement dans la structure de la base de données, une reprise (sans accès direct à la base de données, supposant des connaissances approfondies des bases de données) n'est pas possible. Cela signifie qu'une reconfiguration de ces archives et de ces points de mesure est nécessaire ou que les données doivent être reprises automatiquement par copie d'un projet typeglobal au début de la configuration.

3.3.9.9 Réutilisation de modèles d'impression

Copiez les modèles d'impression souhaités **.rpl* pour les mises en pages ou les **.rp1* pour les gabarits de lignes depuis le répertoire source dans le répertoire *\PRT* du nouveau projet.

3.3.9.10 Réutilisation d'actions globales

Copiez les actions globales souhaitées ou les actions d'arrière-plan **.pas* depuis le répertoire source dans le répertoire *\Pas* du nouveau projet.

3.3.9.11 Réutilisation de fonctions de projet

Copiez les fonctions de projet souhaitées **.fct* depuis le répertoire source dans le répertoire *Library* du nouveau projet. Pour déclarer ces fonctions au projet, activez dans l'éditeur *Global Scripts* la commande de menu *Options→Générer en-tête* . Ceci est décrit de manière plus détaillée dans le chapitre *4.1 Environnement de développement de scripts dans WinCC*.

3.3.9.12 Utilisation de fonctions standard

A la différence des *fonctions de projet*, il n'est pas nécessaire de copier les *fonctions standard*. Ces fonctions sont disponibles immédiatement pour le projet, étant connues de tous les projets WinCC installés sur la station.

3.3.9.13 Réutilisation de la gestion des utilisateurs

La définition des groupes d'utilisateurs, des utilisateurs et des droits d'accès étant intégrée directement dans la structure de la base de données, la réutilisation n'est pas possible. Il est donc nécessaire de procéder à une reconfiguration.
La réutilisation automatique de données par copie d'un projet type global au début de la configuration constitue une exception.

3.3.10 Commande sans souris

Les vues de process se commandent dans la plupart des cas avec la souris dans WinCC. Le clic de la souris est, parmi les modes de dynamisation, l'événement le plus fréquent et connaissant le plus grand nombre de variantes (clic bouton gauche, enfoncer ou relâcher bouton droit). Il existe cependant des systèmes commandés en mode mixte ou uniquement par un clavier. Les entrées sur pupitre opérateur par exemple se font uniquement au clavier.

3.3.10.1 Commande par clavier

La commande par clavier connaît les possibilités d'introduction suivantes:

- Touches de fonctions F1 à F12
- Touches de fonction spéciales (p. ex. touches de fonction de pupitre opérateur (OP) SF10)
- Introductions clavier standard
- Déplacement dans les champs de saisie et commande par touches de sens ou touches spéciales

La configuration des commandes sans souris doit être examinée au cas par cas pour les domaines de configuration suivants :

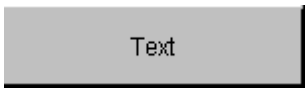
- Touches de commande dans vue de process (p. ex. pour changement de vue)
  - par touches de fonction
  - par touches spéciales
  - par touches standard
- Commande par touches quelconques
- Déplacement dans objets de commande
- Champs de saisie dans vue de process
  - Champs d'entrée/sortie
  - Objets de saisie spéciaux (cases à cocher, etc.)

- Alarm Logging (fenêtre d'alarme)
  - Commande par touches de fonction
  - Commande par touches configurées spécialement

- Tag Logging (fenêtre de courbes ou de tableaux)
  - Commande par touches de fonction
  - Commande par touches configurées spécialement
- Lancement de travaux d'impression par touche
- Connexion au système ou déconnexion (Login/Logoff) par clavier

Les touches de commande sont configurées dans le *style Windows* typique. Vous retrouverez pour cette raison la touche de commande Windows standard dans la palette d'objets. Cette touche peut être complétée à tout moment pour des éléments graphiques supplémentaires.

Touches de commande

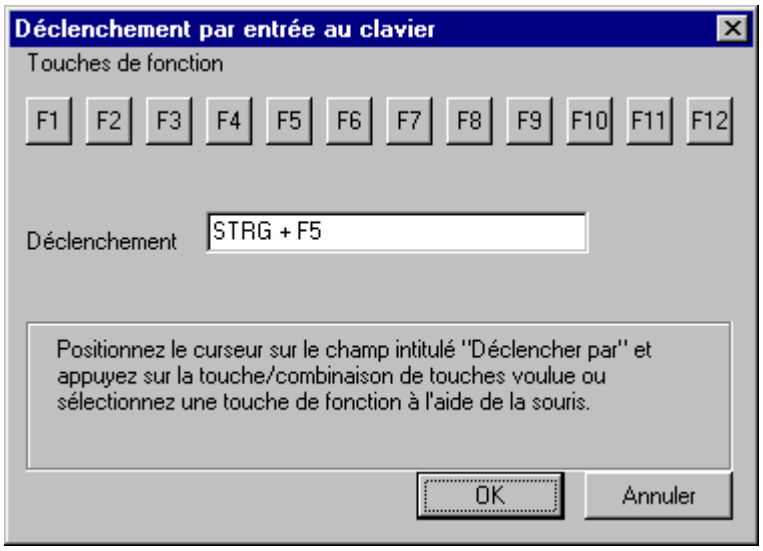


Commande par touches de fonction

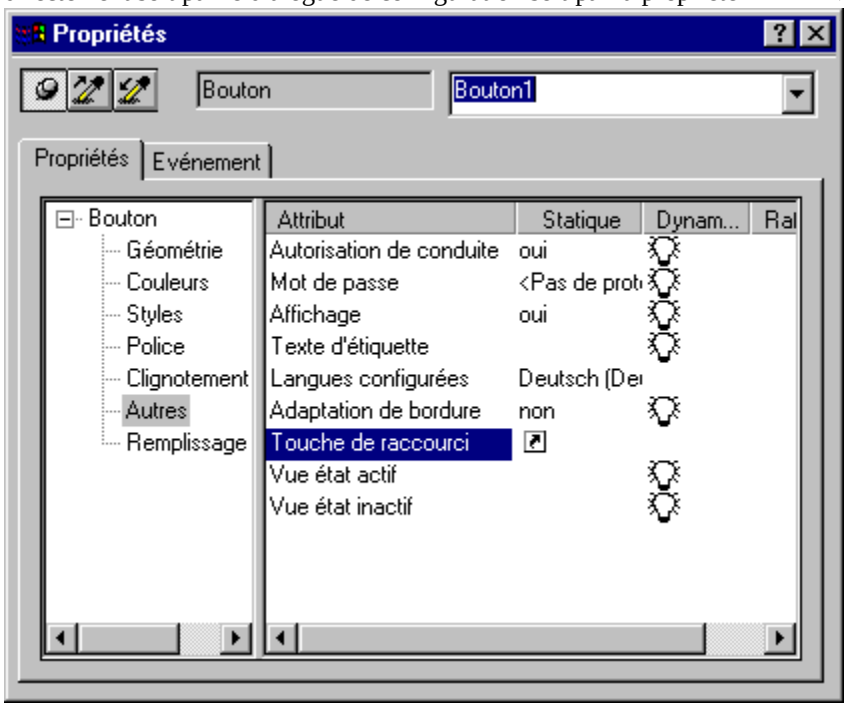
Les touches de fonctions F1 à F12 du clavier standard sont fréquemment utilisées pour la commande clavier (supplémentaire) pour le changement de vue dans la hiérarchie des vues de process. Ces touches de fonction peuvent à tout moment être configurées en 'touches de raccourci' pour les boutons Windows configurés. Une touche de raccourci définit une commande rapide pour la fonction correspondante.

Les touches de fonction mentionnées peuvent être p. ex. mémorisées comme touches de raccourci. Celles-ci sont proposées comme boutons de sélection dans le masque de configuration.

Lorsqu'une combinaison de touches est nécessaire p. ex. la touche MAJ ou la touche CTRL, la séquence de touches doit être entrée directement dans le champ de définition par enfoncement de la touche correspondante (p. ex. MAJ, F2). Il n'est pas nécessaire d'entrer des codes spéciaux.
La combinaison de touches choisie est affichée dans le champ de saisie.

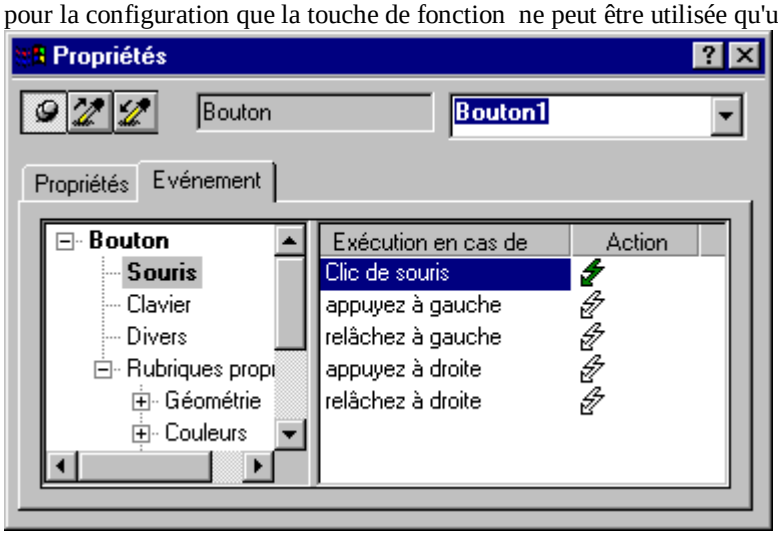


Cette sélection de touches est mémorisée dans les propriétés de l'objet et peut être modifiée directement soit par le dialogue de configuration soit par la propriété *Autres → Touche de raccourci*.



Action pour touche de raccourci

L'action devant être déclenchée par la touche de fonction définie (définition de touche de raccourci) doit être mémorisée dans les événements du bouton Windows. L'événement déclencheur doit être le **clic de la souris**. L'événement est déclenché au relâchement du bouton de la souris, mais uniquement si le pointeur de la souris se trouve sur l'objet au moment de l'enfoncement et du relâchement du bouton. L'action **n'est pas** déclenchée lorsqu'elle n'est pas placée directement derrière l'événement *clic de la souris*, mais par exemple derrière l'événement (semblable) *Appuyez à gauche*! Notez également pour la configuration que la touche de fonction ne peut être utilisée qu'une seule fois dans la vue.



Touches de fonctions spéciales

Lorsque des touches de fonction spéciales p. ex. les touches du terminal opérateur F13, S1, etc., sont utilisées pour manipuler une vue, ces touches doivent être configurées pour former des combinaisons de touches. La touche F13 peut être p. ex. affectée à la combinaison MAJ F1. Les combinaisons de touches choisies sont, outre leur utilisation décrite ci-dessus, définies de manière spécifique au matériel dans la visualisation. Les définitions de touches spéciales diffèrent en fonction du matériel utilisé. Par exemple, un fichier *FI25.key* permet la définition des codes de touches pour les PC industriels. Les codes correspondant à ces touches de fonction sont mémorisés dans ces fichiers spécifiques au matériel. Une fois la définition du clavier du matériel spécifique adaptée - pour chaque fonction de touche, utilisation du code hexadécimal correspondant - et après activation des nouveaux codes de touches, ces touches peuvent être utilisées dans les vues de process pour la commande de vues.

Touches standard

Lorsque l'action ne se réfère pas à l'introduction de touches de fonction, mais à une touche sélectionnée au clavier standard, p. ex. *lettre m*, cette touche est mémorisée comme touche de raccourci dans l'objet *Bouton Windows*.

Commande par touches quelconques

La forme des boutons de commande peut être conçue pour les vues de process. Vous trouverez d'autres touches de fonction p. ex. dans la bibliothèque utilisateur sous *Boutons 3D* ou dans les *objets utilisateur*.
Les objets conçus spécialement à cet effet qui ne sont pas basés sur le bouton Windows ne peuvent être configurés à l'aide de la touche de raccourci. Tous les autres objets doivent être configurés à l'aide de l'*événement bouton* de l'objet pour la commande par touches. Il existe pour cela pour chaque objet les événements de clavier ci-dessous:

- enfoncer
- relâcher

Cet événement de touche doit être disponible pour la configuration éventuelle de la commande clavier. Lorsqu'on utilise des touches prédéfinies issues de la bibliothèque utilisateur, il est par conséquent nécessaire de vérifier si cette touche est adaptée à la commande sans souris. Les boutons de basculement des objets utilisateur, par exemple, ne sont pas toujours déverrouillés pour la commande clavier. Une adaptation peut par conséquent être nécessaire.

Les commandes par touches préconfigurées (p. ex. feuilleter dans la hièrarchie des vues) se trouvent dans les packages optionnels (p. ex. *Basic Process Control - Picture Tree Manager* etc.).

Lorsque l'un de ces objets est utilisé comme élément de commande, la touche permettant le déclenchement est configurée derrière l'événement *Clavier - enfoncez* ou *Clavier - relâchez*. L'action peut être configurée comme *liaison directe* ou comme *action C*.
L'événement de touche déclencheur est soit une

- action de touche quelconque soit une
- touche sélectionnée sur le clavier standard.

S'il s'agit d'une touche quelconque, la *liaison directe* peut être utilisée ; si, par contre, le système doit tester la frappe d'une touche spéciale, une *action C* doit être employée. L'*action C* vérifie le code de touche introduit avant de poursuivre la séquence d'instructions de l'action:

```
if (nChar == S)
    OpenPicture("Start.pdl");
```

Lorsque le programme teste des combinaisons de touches, le premier test de code de clavier doit être suivi d'autres. On utilise pour cela la fonction spéciale „GetAsyncKeyState“ de la manière suivante:

```
#include "apdefap.h"
void OnKeyDown(char* lpszPictureName, char* lpszObjectName, char*
                lpszPropertyName, UINT nChar, UINT nRepCnt, UINT
                nFlags)
{

#pragma code("user32.dll");
int GetAsyncKeyState(int vKey);
#pragma code();

int nRet;

//Interrogation de F1
if (nChar==112) {
   //Si F1 alors interrogation d'actionnement de la touche MAJ
    nRet = GetAsyncKeyState(VK_SHIFT);
   //Bit activé pour touche enfoncée
    if (nRet & 0x8000) {
        // D'autres instructions, déclenchées par la
        // combinaison de touches appropriée
        // peuvent être éditées ici
        }
   //Si F1 alors interrogation d'actionnement de la touche CTRL
    nRet = GetAsyncKeyState(VK_CONTROL);
   //Bit activé pour touche enfoncée
    if (nRet & 0x8000) {
        // D'autres instructions, déclenchées par la
        // combinaison de touches appropriée
        // peuvent être éditées ici
        }
   //Si F1 alors interrogation d'actionnement de la touche ALTGR
    if (nFlags & KF_ALTDOWN) {
        // D'autres instructions, déclenchées par la
        // combinaison de touches appropriée//
        // peuvent être éditées ici
        }
   //Si F1 alors interrogation d'actionnement de la touche ALT
    nRet = GetAsyncKeyState(VK_MENU);
   //Bit activé pour touche enfoncée
    if (nRet & 0x8000) {
        // D'autres instructions, déclenchées par la
        // combinaison de touches appropriée
        // peuvent être éditées ici
        }
    }// Fin d'interrogation du clavier
}
```

3.3.10.2 Déplacement dans les objets de commande (champs de saisie et champ de commande)

Tous les objets pouvant être commandés par la souris peuvent être directement cliqués. La transformation du pointeur indique si l'objet peut être commandé par souris. Comment manipuler ces objets sans souris ?

Curseur - pointeur

Les touches de déplacement permettent en runtime de sauter d'un objet manipulable à un autre. On distingue les :

- objets pilotés par curseur (d'introduction) (objets E/S) et
- les objets commandés par pointeur (de commande)

Les objets E/S sont sélectionnés avec le curseur (touche Tab ou MAJ+Tab).
Tous les éléments de commande (la souris, le clavier ou tout autre mode de commande) peuvent être intégrés en un groupe à commande par le pointeur. Les champs d'E/S peuvent être intégrés aussi bien dans un groupe à commande par pointeur que dans un groupe à commande par curseur.

Séquence TAB

La *Séquence TAB* (via *Editer →Séquence TAB →Curseur ou →Pointeur*) permet de définir dans quel ordre les objets manipulables seront sélectionnés avec la touche TAB au runtime. L'objet actuellement sélectionné peut être visualisé au runtime. Ce curseur est le curseur de runtime et peut être désactivé (*Propriétés - Ordinateurs →Graphics Runtime).* Lorsqu'ils sont sélectionnés, les boutons de style Windows sont toujours marqués par un rectangle en pointillé à leur activation.

Le déplacement dans les éléments manipulables dépend du paramétrage de Graphics Runtime (*Propriétés - Ordinateurs →Graphics Runtime).*

Déplacement	Touches standard	Paramétrage touches
vers le haut, vers le bas vers la gauche, vers la droite	Touches de flèche ou Tabulateur (suivant) ou MAJ Tabulateur (précédent)	Autre paramétrage des touches dans *Propriétés - Ordinateurs →Graphics Runtime→Commande curseur-Touches*
Curseur - pointeur	Pointeur	Basculement entre les deux types de curseur par touches de raccourci (*Propriétés - Ordinateurs →Graphics Runtime→Touches de raccourci)* ou définition de touches spécifiques (avec la fonction interne *SetCursorMode*)
Déplacement dans tableau (groupe curseur)	déplacement normal, mode ligne lorsque le curseur a atteint la fin du groupe de curseur, il s'arrête à cette position.	Modification du comportement par *Propriétés - Ordinateurs →Graphics Runtime →Commande du curseur - Comportement*

Champs d'entrée/sortie

Les champs d'E/S configurés peuvent être manipulés directement au clavier dès leur sélection : vous pouvez y entrer immédiatement vos nouvelles donnés. Un champ de sortie uniquement (*Propriétés →Entrée/Sortie →Type de champ → Sortie)* n'est pas manipulable.
La validation de la saisie s'effectue, avec la touche ENTREE, en fonction de la propriété configurée (*Propriétés →Entrée/Sortie →Valider en quittant*). La touche d'échappement (ESC) met par contre fin à l'introduction sans modification de la valeur.

Autres objets de saisie

Outre les champs typiques d'entrée de valeurs analogiques, les applications Windows connaissent d'autres possibilités de saisie. Ces objets spéciaux se trouvent dans la palette d'objets sous *Objets Windows*

- Case à cocher
- Case d'option

L'activation des champs de sélection des cases à cocher ou des cases d'option se fait avec la touche d'espacement, le déplacement à travers les divers éléments de la boîte avec les touches vers le haut/vers le bas (p. ex. touches de sens). Ceci est l'affectation de touches standard.
L'**objet liste de textes** est un autre objet de saisie. Une sélection peut être effectuée dans une liste affichée en fonction des entrées configurées:

La commande peut également s'effectuer au clavier. La mémorisation d'une configuration spéciale n'est pas nécessaire pour l'affectation des touches.
La liste s'ouvre à l'aide de la touche ENTREE ; es déplacements dans la liste s'obtiennent avec les touches de déplacement vers le haut/vers le bas ; la validation de la sélection courante se fait avec la touche ENTREE.

D'autres objets de saisie peuvent être utilisés dans WinCC à l'aide de contrôles OCX. Leur manipulation et leur configuration sont cependant fonction des événements et des propriétés disponibles qui sont définis pour un objet. Ceci doit être élucidé dans chaque cas.

3.3.10.3 Touches de fonction d'Alarm Logging comme touches de barre d'outils

Dans les fenêtres d'alarme, divers boutons de commande sont placés dans la barre d'outils et manipulés en standard avec la souris.

Les manipulations les plus fréquentes dans une fenêtre d'alarme sont

- Sélection d'une alarme pour l'acquittement
- Déplacements vers le haut/vers le bas dans la liste d'alarmes
- Feuilleter dans la liste d'alarmes

Nota:
A l'ouverture de la fenêtre d'alarme ou une autre commande, la commande par boutons doit être disponible dans la fenêtre d'alarme (fenêtre d'application) et pas dans la fenêtre principale. Les touches de fonction agissent, en fonction de la fenêtre courante, soit sur la barre de touches de fonction de la fenêtre principale soit sur les touches de commande de la fenêtre d'alarme.

Ceci s'obtient par exemple en activant le focus de commande courant dans cette zone de fenêtre. Le focus se positionne normalement en cliquant dessus avec la souris.
Le focus peut être positionné au clavier dans la fenêtre d'alarme par trois voies configurables :

- changement de fenêtre avec touche de raccourci
- positionnement du focus par une touche de commande ou
- positionnement direct du focus sur un élément défini dans la fenêtre d'alarme à la sélection de vue.

Le basculement dans la fenêtre d'alarmes à l'aide d'une commande rapide (touche de raccourci) pouvant être utilisé pour tous les changements de fenêtre, c'est-à-dire aussi pour les fenêtre de courbes p. ex., se règle dans les paramètres de lancement de Graphics Runtime. La combinaison de touches (p. ex. CTRL W) est par exemple entrée dans *Propriétés - Ordinateurs→Graphics Runtime →Touches de raccourci →Changer de fenêtre* .

Une fois la fenêtre d'alarmes sélectionnée, cette séquence de touches permet de commander directement les boutons de la barre d'outils.

Le positionnement direct du focus de commande sur la fenêtre d'alarme s'obtient par contre avec la fonction interne *Set_Focus*. Une *action C* programmée pour une commande par touche ou pour une sélection de vue (*Objet vue→Evénement→Autres →Sélections de vue*) peut par conséquent influer sur la commande de la fenêtre d'alarme.

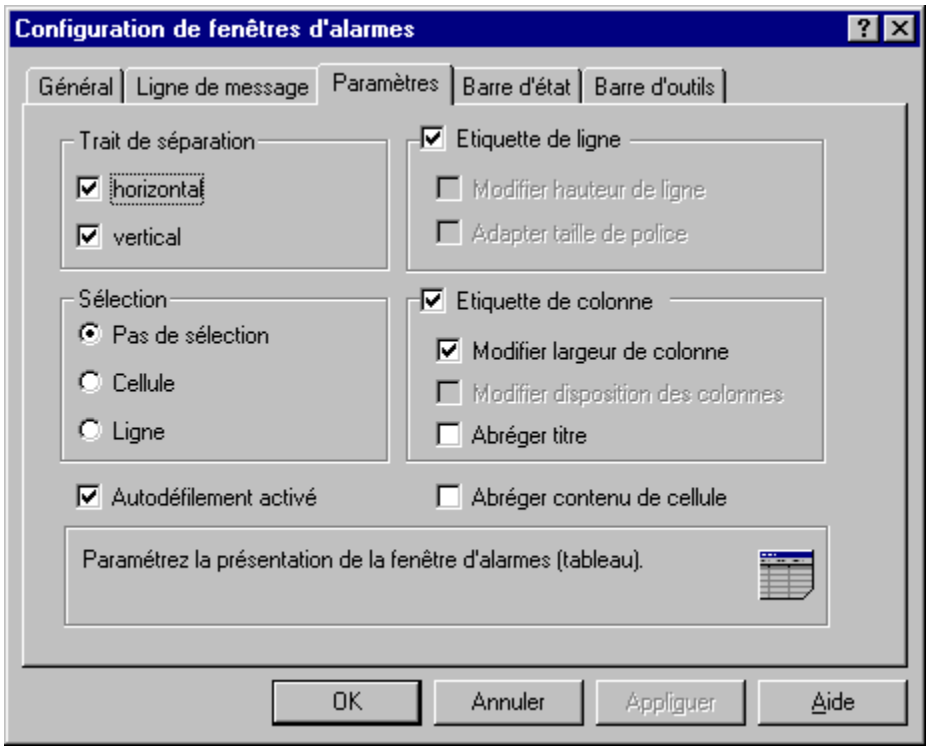
Vous trouverez les fonctions de focus de vue sous *Fonctions internes →graphics →Set →focus.* La fonction suivante est appelée par exemple pour activer le focus de commande :

```
Set_Focus(nom de vue, nom d'objet);
```

Entrez comme paramètres le nom de la fenêtre principale (nom de vue) et de la fenêtre d'application (nom d'objet).

La sélection d'une alarme dans la fenêtre d'alarmes se fait en sélectionnant la ligne d'alarme. A la sélection de la fenêtre d'alarmes, le curseur courant se trouve sur l'alarme la plus récente (dernière alarme dans la vue d'alarmes). La possibilité de sélectionner une alarme dans la fenêtre ou de feuilleter les alarmes dépend de l'activation/la désactivation du défilement.

L'activation/la désactivation du défilement de l'image peut être déclenchée par un bouton spécifique dans la barre d'outils ou paramétrée directement dans la configuration du modèle de fenêtre d'alarmes. L'activation du défilement de l'image à l'affichage d'une vue se paramètre dans l'éditeur *Alarm Logging →Modèles de fenêtres d'alarmes →Paramètres→ Autodéfilement activé.*

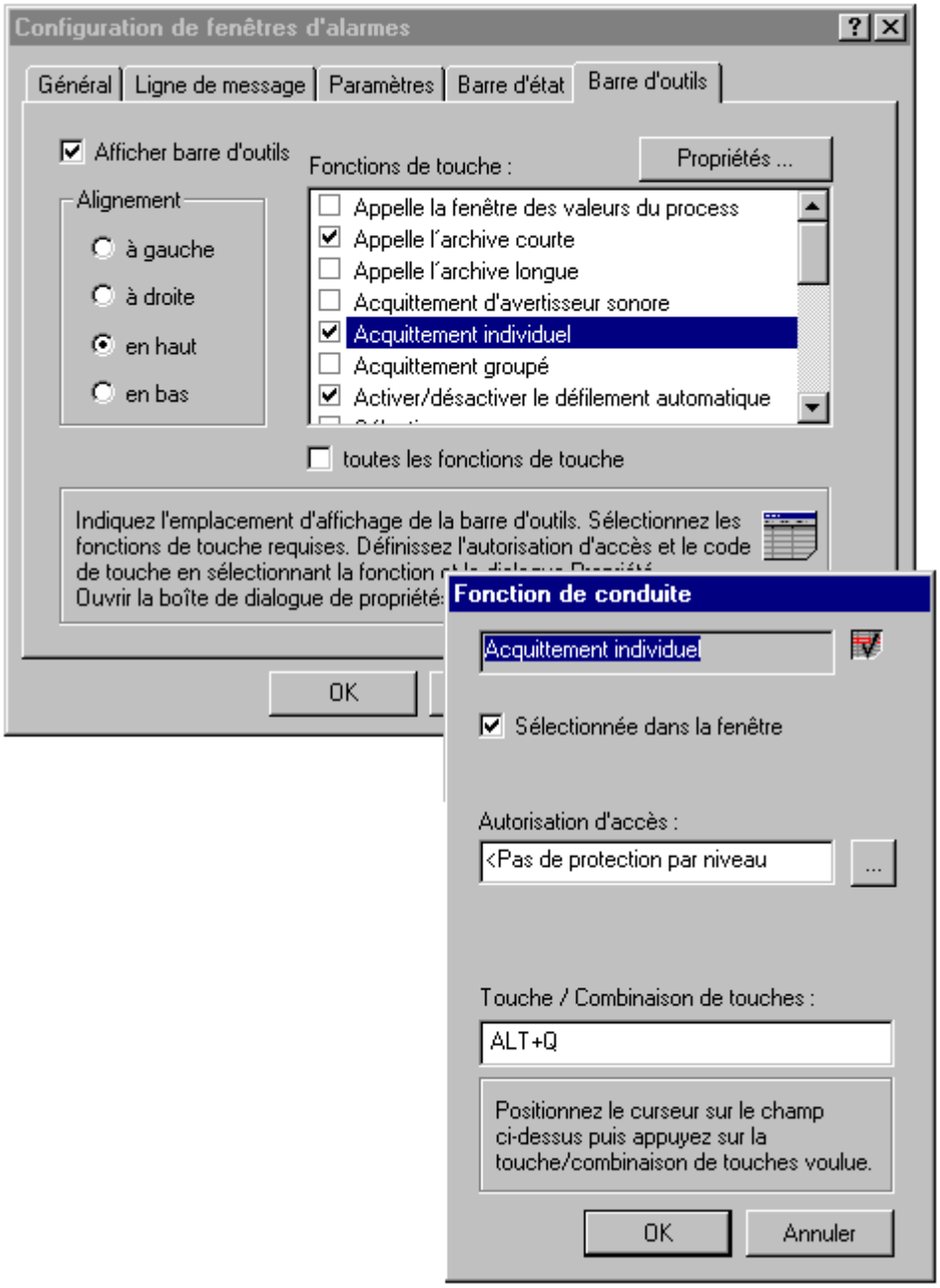


Lorsque le défilement de l'image et le focus de commande sont activés dans la fenêtre d'alarmes, les déplacements peuvent s'effectuer comme suit :

Déplacement	**Touches standard**	**Paramétrage touches**
Déplacements vers le haut/vers le bas dans la liste d'alarmes	Touches de sens	Lignes d'alarmes
Début, fin de la liste d'alarmes	Touche Pos1, Fin	Début ou fin de la vue d'alarmes
Feuilleter	Touches Feuilleter	plusieurs lignes d'alarmes

Outre la commande standard avec la souris, on peut également définir une commande par touche de fonction pour activer les boutons de commande de la barre d'outils, p. ex. pour l'acquittement de l'alarme sélectionnée.
Pour chaque touche visualisée dans la barre d'outils (*Alarm Logging →Modèles de fenêtre d'alarmes →Barre d'outils →cochée et sélectionnée →Propriétés*), une commande par touche peut être mémorisée dans les propriétés.

Par exemple:



Ces opérations de configuration permettent d'affecter une combinaison de touches à tous les boutons de commande utilisés pour la fenêtre d'alarmes. Il faut par conséquent définir en outre une commande de bouton dans la fenêtre d'alarmes.

3.3.10.4 Boutons de barre d'outils Alarm Logging spécifiques au process

La totalité des boutons de la barre d'outils sont proposés par WinCC et ne peuvent être modifiés. Si une disposition de bouton est prescrite pour le process à configurer, la barre d'outils WinCC doit être désactivée (vous n'avez donc plus de barre d'outils). Vous devez configurer vous-même les boutons correspondants selon les souhaits du client (p. ex. leur affecter des vues).

Il faut cependant également configurer la fonctionnalité des divers boutons comme action. Dans l'*action C* de l'événement correspondant (p. ex. enfoncer bouton), la *fonction standard* associée doit être sélectionnée dans l'arbre des fonctions.

Les fonctions disponibles pour la commande de boutons se trouvent dans les *fonctions standard* → *alarm.* A chaque bouton de la barre d'outils correspond une fonction associée dans la liste. La fonction ci-dessous est par exemple appelée pour le bouton d'acquittement:

```
OnBtnSinglAckn(nom de fenêtre);
```

Le nom du *modèle de fenêtre d'alarmes* doit être entré comme paramètre.
Ces actions peuvent également être utilisées avec la souris pour les boutons de commande configurés par l'utilisateur.
Vous trouverez quelques exemples de ces boutons dans les packages optionnels du système d'alarmes (p. ex. *Basic Process Control* - Acquittement de klaxon, etc.).

3.3.10.5 Boutons de fonction Tag Logging comme boutons de barre d'outils

Dans les fenêtres de courbes et de tableaux représentant des valeurs de mesure (nommées 'affichages de tendance'), des boutons manipulables en standard avec la souris sont placés dans la barre d'outils.

Les principales commandes d'une fenêtre de courbes sont

- Feuilleter dans les valeurs de mesure (axe des temps)
- Sélection d'une période
- Sélection de courbes
- Commande de ligne de lecture

Après affichage de la fenêtre de courbes, la courbe courante est affichée en fonction de la configuration.

Nota:
A l'affichage de la fenêtre de courbes, les boutons de commande doivent se trouver dans la fenêtre de courbes (fenêtre d'application) et pas dans la fenêtre principale. Selon la fenêtre dans laquelle les boutons de commande sont disponibles, ces commandes (touches de fonction) agissent sur la *barre de boutons de fonction* de la fenêtre principale ou sur les commandes par boutons mémorisés de la fenêtre de courbes ou de tableaux.

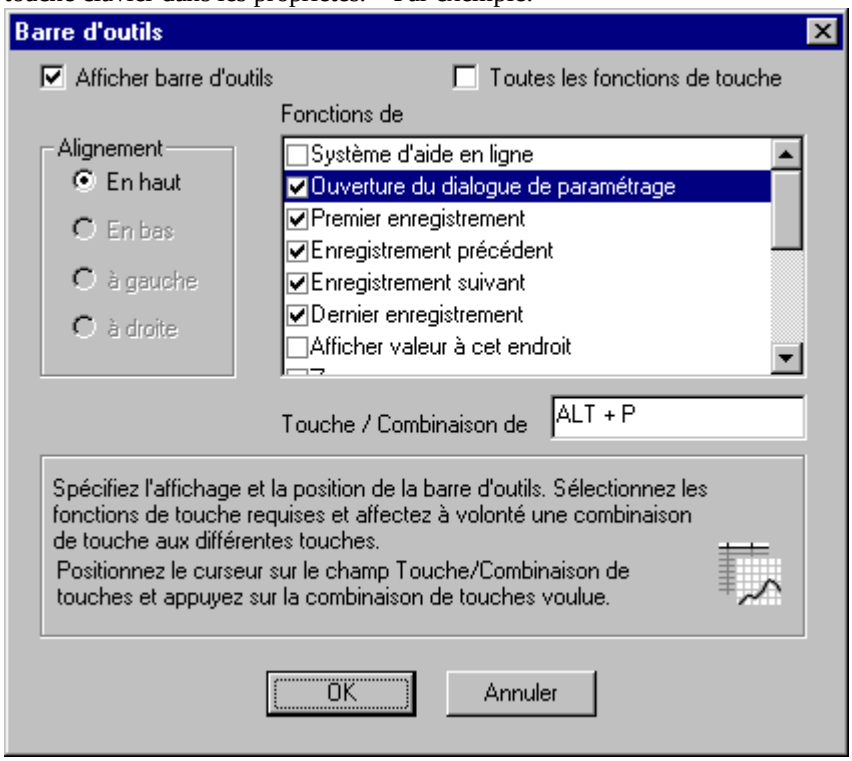
Ceci s'obtient par exemple en activant le focus de commande courant dans cette zone de fenêtre. Le focus se positionne normalement en cliquant dessus avec la souris.
Le focus peut être positionné au clavier dans la fenêtre de courbes ou de tableaux par une des variantes configurables suivantes:

- changement de fenêtre avec touche de raccourci
- positionnement du focus par une touche de commande ou
- positionnement direct du focus sur un élément défini dans la fenêtre d'alarme à la sélection de vue.

La réalisation de ces variantes a été décrite au chapitre Alarm Logging.

Outre la commande standard avec la souris, on peut également définir des touches de fonction pour activer les boutons de commande de la barre d'outils, comme p. ex. pour la sélection d'une période. Les touches sont occupées en standard par les fonctions F1 à F10.

Pour chaque bouton visualisé dans la barre d'outils (*Tag Logging →Modèle de fenêtre de courbes →Modèle →Barre d'outils →Configurer...*), l'utilisateur peut mémoriser sa propre commande par touche clavier dans les propriétés. Par exemple:



Ces opérations de configuration permettent d'affecter une combinaison de touches à toutes les touches de commande utilisées pour la fenêtre de courbes ou la fenêtre de tableaux. Il faut donc définir en plus une commande de touche pour les fenêtres de courbes et de tableaux.

Dans la fenêtre de courbes, les touches standard suivantes peuvent être utilisées après activation de la touche de fonction correspondante:

Déplacement	**Touches standard**	**Paramétrage touches**
Ligne de lecture	Touches de sens	Déplacer la ligne de lecture vers la gauche ou vers la droite
Zoom	Sélectionner section pour zoom	Régler le mode d'introduction de substitution pour la souris et l'activer (voir paramétrage système) INS et les touches de sens permettent de définir la section pour le zoom
Dialogues p. ex. sélection de variables d'archive	Touche Tab	pour déplacement dans les champs de saisie
	Touches de sens	déplacement dans la sélection de variables ou la sélection d'onglets;
	Touche +(Touche -)	affichage ou fermeture de l'arbre des variables d'archive ;
	Touche Espacement	Sélection ou annulation de sélection
	Touche ENTREE	Validation et fermeture de boîte de dialogue
	Touche Echap.	Interrompre boîte de dialogue.

Boutons de barre d'outils configurés de manière spécifique au process dans Tag Logging

La totalité des boutons de la barre d'outils sont proposés par WinCC et ne peuvent être modifiés. Si une disposition particulière des boutons est prescrite pour le process à configurer, la barre d'outils WinCC doit être désactivée et vous devez configurer vous-même les boutons correspondants Vous devez configurer vous-même les boutons correspondants selon les souhaits du client (p. ex. leur affecter des vues).
Il faut cependant également configurer la fonctionnalité des divers boutons comme action. Dans l'*action C* de l'événement correspondant (p. ex. enfoncer bouton), la *fonction standard* associée doit être sélectionnée dans l'arbre des fonctions.

Les fonctions disponibles pour la commande de boutons se trouvent dans les *fonctions standard* → *TAGLOG (*ou en plus → *TEMPLATE* pour les fonctions de tableau . A chaque bouton de la barre d'outils correspond une fonction associée dans la liste. La ligne de lecture est par exemple appelée par la fonction suivante:

```
TlgTrendWindowPressLinealButton(nom de fenêtre);
```

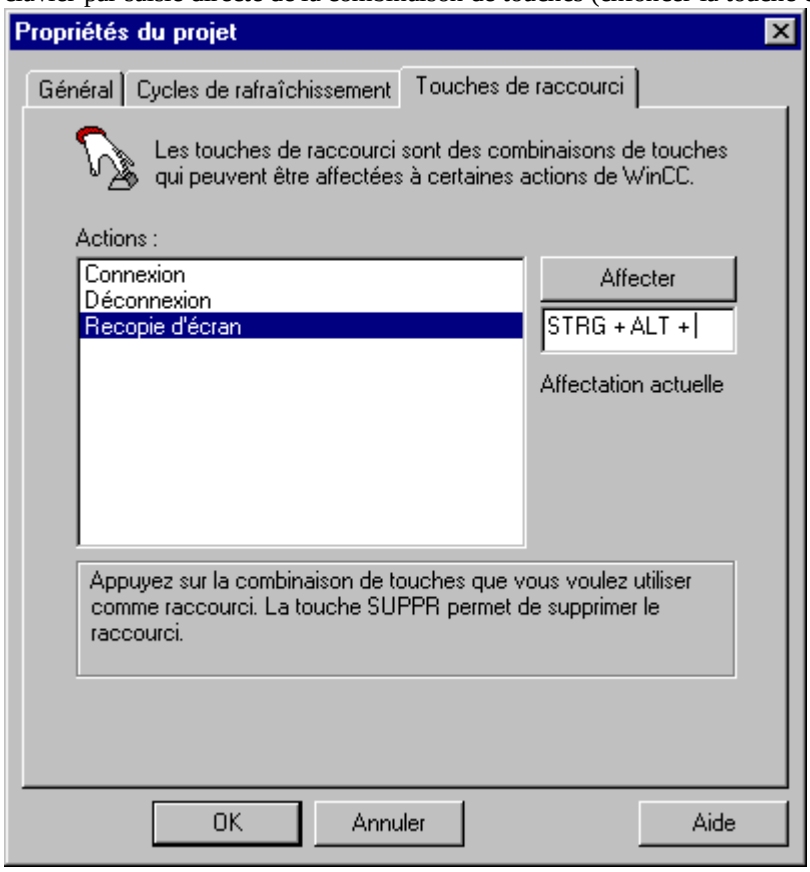
Le nom du *modèle de fenêtre de courbes* doit être entré comme paramètre.

Ces actions peuvent également être utilisées avec la souris pour les boutons de commande configurés par l'utilisateur.

3.3.10.6 Déclenchement de travaux d'impression

Un travail d'impression peut être lancé de plusieurs manières. Dans le *Control Center* par exemple, l'impression est activée directement par sélection dans la liste des travaux d'impression. Il est cependant possible de créer également dans la vue de process un bouton permettant de lancer le travail d'impression.
Pour les listes d'alarmes, ce bouton existe déjà dans la barre d'outils et peut être activé par une touche de fonction ou une touche configurée spécialement à cet effet comme décrit dans les pages précédentes.

Une recopie d'écran peut être activée dans chaque vue par une touche de raccourci. Cette touche de raccourci se paramètre au niveau global dans les propriétés de projet. Appelez pour cela *Propriétés projet →Touche de raccourci →Recopie d'écran* dans *Control Center* et définissez le raccourci clavier par saisie directe de la combinaison de touches (enfoncer la touche correspondante du clavier).



Lorsqu'une touche a été configurée spécialement pour l'impression d'un travail d'impression défini dans une vue de process, l'impression doit être lancée par une *action C*. Comme il a été indiqué au début de ce chapitre, la touche peut être commandée p. ex. par une touche de fonction (voir touches de raccourci) ou par une commande clavier (p. ex. touche D). L'*action C* doit être par conséquent configurée pour l'événement associé (p. ex. *clic de souris* ou *Touche - appuyez clavier*). WinCC met cette fonctionnalité à disposition sous *Fonctions standard →Report →ReportJob* .

```
ReportJob("Documentation Control Center","PRINTJOB");
```

La fonction prend le nom du travail d'impression. Le nom de méthode est 'Impression sur imprimante' ou 'Aperçu'.

3.3.10.7 Connexion au système ou déconnexion

Outre les touches de raccourci configurables pour l'opération de connexion ou de déconnexion (Login, Logout) il est également possible de configurer une touche permettant d'appeler la boîte de dialogue de connexion. La déconnexion par touche est également possible. Il faut pour cela configurer un bouton spécifique pouvant être commandé par exemple par clic de la souris et aussi par commande clavier. La commande par touche de fonction peut également être paramétrée dans la propriété de touche de raccourci de la touche. Les différentes variantes de commande par touche sont décrites au début du présent chapitre. La fonction devant être utilisée pour la connexion et la déconnexion est une fonction d'application WinCC. L'utilisation de cette fonction doit être configurée comme *action C*. Liez par exemple l'*action C* à l'événement *clic de la souris* ou l'événement *Touche - appuyez*.
La fonction ci-dessous est utilisée pour la déconnexion:

```
#pragma code ("useadmin.dll")
#include "PWRT_api.h"
#pragma code ()

PWRTLogout();
```

La connexion au système se fait à l'aide de la fonction d'application C WinCC *PWRTLogin()*.
Exemple d'utilisation de ce cette fonction:

```
#pragma code ("useadmin.dll")
#include "PWRT_api.h"
#pragma code ()
PWRTLogin(c);
```

La boîte de dialogue affichée peut être commandée avec les touches standard:

Déplacement	**Touches standard**	**Paramétrage touches**
Champs individuels de saisie	Touche tabulateur (vers l'avant) ou MAJ et touche tabulateur (vers l'arrière) Touches de sens	Déplacer la ligne de lecture vers la gauche ou vers la droite
Validation (OK)	Touche ENTREE	Quitter la boîte de dialogue et valider la saisie;
Annuler	Touche Echap.	Annuler boîte de dialogue ou saisie

3.3.11 Technique des composants de vue

La technique des composants de vue est une approche essentielle pour la configuration rapide et simple et pour la réutilisabilité et maintenabilité des composants de vue configurés.

Une boîte de process configurée sera par exemple utilisée pour plusieurs composants de process semblables (p. ex. électrovannes ou régulateurs).

La fenêtre de vue peut, une fois configurée, être réutilisée pour les composants de régulation à visualiser et à commander dans le projet, d'après les principes ci-dessous:

- Copie d'une fenêtre et création de nouvelles liaisons pour les champs de variables
- Utilisation d'une fenêtre dont les champs de variables sont affectés à l'appel (liaison indirecte)
- Utilisation d'objets utilisateur avec prototypes et objets créés à partir de ceux-ci
- Création de vues prototypes et liaison de celles-ci
- Création de composants OCX et liaison de ceux-ci comme objets OCX WinCC

Comparaison des diverses techniques

Ces techniques sont très différentes quant à la manière de les utiliser, la complexité de leur configuration et leurs possibilités ; nous commencerons par conséquent par une comparaison des diverses possibilités.

Type	Avantage	Inconvénient
Copie de fenêtres	simplicité	toutes les liaisons d'objets doivent être modifiées Les modifications dans la construction de la vue entraînent des adaptations ultérieures lourdes
Fenêtre avec liaison indirecte	• fenêtre construite une seule fois avec des *actions C simples* • réutilisation sans copie de la fenêtre type	les modifications apportées à la construction des vues entraînent des adaptations lourdes
Objets utilisateurs	Objets construit une seule fois avec liaison par assistants Dynamic-Wizard existants	• Les modifications dans la construction de la vue entraînent des adaptations lourdes, c'est-à-dire une nouvelle création de vue • pas de modification centralisée possible
Vues prototypes	• Objet construit une seule fois • modification centralisée possible	(bonnes) connaissances de C nécessaires
OCX	• faciles à lier dans la configuration de WinCC comme objet dans une vue • une modification ultérieure de l'objet OCX n'entraîne aucune adaptation sur les objets créés sauf pour les	doivent être créés par programmation (C++,VB 5) ; ne peuvent être créés par configuration WinCC.

Type	Avantage	Inconvénient
	modifications des propriétés d'objet • méthode très performante • autres possibilités graphiques • Achat ultérieur de nouveaux objets (p. ex. composants PCS7)	

Si quelques composants de vue simples seulement sont utilisés dans le projet, une des premières variantes suffira au technicien de configuration. Celles-ci peuvent être utilisées sans grande initiation. L'objet utilisateur s'avère particulièrement bien adapté aux objets simples de faible et moyenne complexité et à la liaison à des variables. Si cet objet nécessite plusieurs modifications ultérieures prévisibles, une initiation au concept du prototype de vue sera rentable.
Lorsque les composants graphiques sont complexes ou lorsqu'une puissance de traitement importante est nécessaire, la technique des OCX sera recommandée. Les objets OCX disponibles seront de plus en plus nombreux à l'avenir pour les divers métiers.

Nous exposerons ci-dessous les principaux types de configuration de composants de vue et leur utilisation dans les vues de process. Vous pouvez ainsi vous faire une idée des différentes variantes disponibles et de leurs domaines d'application dans vos projets.

3.3.11.1 Boîte de process comme composant de vue

Des boîtes d'information spécifiques sont incrustées dans les vues de process pour représenter les états courants d'un composant de process (régulateurs, électrovannes, moteurs, etc.) ou pour prescrire des valeurs de consigne. Ces boîtes de process contiennent typiquement à la fois les états courants (valeurs réelles) et les valeurs de consigne pouvant être entrées par l'opérateur privilégié.

Création d'une boîte d'information

Cette boîte d'information se crée sous la forme d'une fenêtre dont les composants sont liés aux variables (de process) correspondantes.

Etape	Type	Configuration
1	Structures de données	Définissez dans le composant de vue les structures de données à employer dans la gestion des variables, p. ex. moteur avec valeur réelle, valeur de consigne, commutateur marche/arrêt.
2	Composant de vue	Configurez dans *Graphics Designer* une vue montrant les états du composants de process, p. ex. de bargraphes et des champs d'E/S ainsi que des boutons de commande. La taille de la fenêtre (propriété objet de vue - Dimension x et Dimension Y) doit correspondre aux dimensions à obtenir pour la fenêtre de vue.
3	Définition de variables	Définissez les variables (de process) dans la gestion des variables; p. ex. moteur_T01 du type de données (structure) Type de moteur qui sera utilisé dans la boîte de process..
4	Liaison de variables	Dynamisez maintenant les composants de vue, p. ex. champs d'E/S, bargraphes, etc., par création de liaisons aux variables (de process) correspondantes.
5	Fenêtre de vue	Créez un objet de fenêtre dans la vue de process et liez cet objet au contenu de fenêtre configuré par les opérations 2 à 4 via la propriété Nom de fenêtre de vue.

Etape	Type	Configuration
6	Paramétrage des propriétés	L'objet de fenêtre ne doit pas être encore affiché à la visualisation de la vue. La propriété *Affichage* doit être par conséquent paramétrée sur Non de manière statique. Il faut encore définir l'aspect de la fenêtre avec ses boutons Windows ainsi que le titre, etc. dans les propriétés de la fenêtre.
7	Appel de la fenêtre	Cette fenêtre doit être p. ex. affichée par activation d'un bouton de commande ou par commande du composant de process lui-même. Configurez une touche de commande qui sera liée à l'affichage de l'objet de fenêtre (par *liaison directe* p. ex.)

Cet objet de fenêtre, le contenu de la fenêtre et l'élément de commande servant à l'appel correspondant de la fenêtre (touche de commande) peuvent être réutilisés sous une forme semblable pour d'autres composants de process. Il faut pour cela copier l'objet de fenêtre, le composant de vue et la touche de commande. Les références doivent être adaptées. Pour la copie, l'objet de fenêtre et la touche de commande peuvent être placés dans la bibliothèque graphique (p. ex. bibliothèque de projet) à l'aide du 'Glisser-Déposer'.

Adaptation des composants de vue

Les opérations suivantes doivent par conséquent être effectuées pour l'utilisation du composant de vue créé:

Etape	Type	Configuration
1	Variables de process	Définissez une nouvelle variable de process, p. ex. moteur_T02 pour la structure de données définie.
2	Copie du composant de vue	Créez une copie du contenu de la fenêtre de vue (Moteur02.PDL) et modifiez toutes les références mémorisées à demeure (p. ex. à la place de Valeur réelle.Moteur_T01 Valeur réelle.Moteur_T02).
3	Copie de la fenêtre	Créez une copie de l'objet fenêtre dans la vue de process cible (par Glisser-Déposer à partir de la bibliothèque graphique). Adaptez la référence au contenu de la fenêtre dans *Propriété →Nom de vue* (Moteur02.PDL).
4	Copie du bouton de commande	Créez une copie du bouton de commande dans la vue de process cible (par Glisser-Déposer à partir de la bibliothèque graphique). Adaptez la référence au nouvel objet de la fenêtre de vue dans la *liaison directe* (*Objet →Fenêtre2 →Affichage)*.

De cette manière, les fenêtres d'une vue et leur contenu peuvent être créés pour chaque composant de process et réutilisés par simple copie. Comme on le voit, le travail à effectuer consiste à adapter les références mémorisées à demeure, pour le contenu de fenêtre par exemple. Il est pour cette raison possible de réaliser plus simplement la réutilisabilité par l'adressage indirect. Le travail d'adaptation doit être réduit au minimum.

Alternative : le composant de vue peut être également configuré sans liaison de fenêtre de vue. Cela signifie que le composant de vue est configuré comme objet non affichable dans la vue de process. Ceci présente cependant pour la modification du composant l'inconvénient considérable qu'une modification doit être effectuée dans toutes les vues utilisant ce composant de vue.

3.3.11.2 Composant de vue avec adressage indirect

Jusqu'alors, les divers éléments du composant de vue étaient liés aux variables (de process) correspondantes. Le composant conçu peut être utilisé de manière beaucoup plus souple si la liaison est déterminée dynamiquement au runtime et non pas par une configuration fixe. Cette liaison dynamique de variables (de process) est réalisée par adressage indirect des divers éléments du composant de vue. Cela signifie que la liaison aux variables (de process) n'est pas établie directement, mais avec le *Conteneur* qui portera le nom courant de la variable (de process) correspondante au runtime.

Le caractère adaptatif et réutilisable d'un composant peut être ainsi considérablement simplifié.
La configuration s'effectue de manière analogue à la procédure décrite préalablement. Les étapes sont les suivantes:

Etape	Type	Configuration
1	Définition de données	Définissez les données à utiliser dans le composant de vue dans la gestion des variables, p. ex. Moteur001_ Valeur réelle, Moteur001_Valeur de consigne, Moteur001_Commutateur, d'une part, et définition des conteneurs de nom pour les divers éléments devant être utilisés dans le composant de vue, p. ex. Nom_Valréelle, Nom_Valcons, etc. d'autre part. Ces variables s'initialisent avec un nom, p. ex. Moteur001_Valcons.
2	Composant de vue	Configurez dans *Graphics Designer* une vue montrant les états du composants de process, p. ex. de bargraphes et des champs d'E/S ainsi que des boutons de commande. La taille de la fenêtre (propriété objet de vue - Dimension x et Dimension Y) doit correspondre aux dimensions à obtenir pour la fenêtre de vue.
3	Liaison de variables	Dynamisez maintenant les composants de vue, p. ex. champs d'E/S, bargraphes, etc., par création de liaisons aux variables de conteneurs correspondantes qui contiennent les noms des variables associées. A la liaison, il faut indiquer que la variable n'est que le nom de la variable (de process) proprement dite. Il faut pour cela faire une croix pour chaque variable dans la colonne *Adr.indir.*.
4	Fenêtre de vue	Créez un objet de fenêtre dans la vue de process et liez cet objet au contenu de fenêtre configuré par les opérations 2 à 3 via la propriété Nom de fenêtre de vue.
5	Paramétrage des propriétés	L'objet de fenêtre ne doit pas être encore affiché à la visualisation de la vue. La propriété *Affichage* doit être par conséquent paramétrée sur Non de manière statique. Il faut encore définir l'aspect de la fenêtre avec ses boutons Windows ainsi que le titre, etc. dans les propriétés de la fenêtre.
6	Appel de la fenêtre	Cette fenêtre doit être p. ex. affichée par activation d'un bouton de commande ou par commande du composant de process lui-même. Configurez un bouton de commande qui sera lié à l'incrustation de l'objet de fenêtre (par *liaison directe* p. ex.).
7	Bibliothèque graphique	L'objet de fenêtre et le bouton de commande peuvent être placés dans la bibliothèque à l'aide du Glisser-Déposer pour leur réutilisation.

3.3.11.3 Objets utilisateur

Des composants de vue facilement réutilisables peuvent être créés à l'aide d'objets utilisateur et des assistants Dynamic-Wizard correspondants. La copie créée du composant de vue peut être liée, par configuration simple avec l'assistant, aux variables (de process) courantes associées.
Un objet utilisateur est un objet graphique créé par un concepteur (p. ex. combinaison de plusieurs objets) dont les nombreuses propriétés et les événements sont limités aux principales propriétés et événements dans un dialogue de configuration. Cet objet utilisateur est dynamisé comme prototype à l'aide de l'assistant correspondant.

La séquence opératoire est la suivante:

Etape	Type	Configuration
1	Structures de données	Définissez les structures de données à employer dans le composant de vue dans la gestion des variables.
2	Composant de vue	Configurez dans Graphics Designer un objet utilisateur ayant les propriétés définies par l'utilisateur

Un objet utilisateur est créé à partir d'un groupe d'objets WinCC. Pour le moment, aucune dynamisation n'est configurée pour ces objets. Tous les objets devant être regroupés dans l'objet utilisateur sont sélectionnés et le dialogue de configuration de l'objet utilisateur est appelé.

Dans ce dialogue, toutes les propriétés des objets sont déclarées être des propriétés de l'objet utilisateur qui seront plus tard dynamisées. Les propriétés de base d'un objet (p. ex. position et taille) ont déjà été définies pour l'objet utilisateur.
Chacune des propriétés des objets regroupés peut être sélectionnée dans le dialogue et ajoutée par Glisser-Déposer au nouvel objet utilisateur comme *propriété ou événement défini par l'utilisateur*.

Un nouveau nom d'attribut (indépendant de la langue) et le nom de propriété dépendant de la langue (p. ex. configuration en anglais) peuvent être attribués par l'utilisateur à chacune de ces propriétés. Ceci permet de ne montrer qu'un faible nombre seulement de propriétés et d'événements (à dynamiser). Ceci permet de ne montrer qu'un faible nombre seulement de propriétés et d'événements (à dynamiser). Toutes les autres propriétés et tous les autres événements seront cachés.

L'objet utilisateur créé doit être maintenant dynamisé. Un assistant est pour cela disponible:

Etape	Type	Configuration
3	Dynamisation	Appelez l'assistant Dynamic-Wizard *Dynamisation prototype*. Liez selon un schéma prototypique chaque propriété de l'objet ou champ correspondant de la structure définie. Le *champ de structure* est sélectionné pour la connexion avec le browser de variables. L'assistant mémorise cependant uniquement le nom du champ de la structure pour la propriété liée (p. ex. 'value'). Chaque propriété doit être liée séparément. Cet objet est maintenant un objet dynamisé mais qui n'a été lié que d'après un schéma prototypique et qui n'a pas encore de vie au runtime. Il ne peut par conséquent pas être rafraîchi au runtime.
4	Rangement dans la bibliothèque graphique	Rangez ce prototype comme objet dans la bibliothèque graphique

L'objet utilisateur prototype est rangé dans la bibliothèque graphique p. ex. pour une réutilisation multiple. Les instruments ou indicateurs à index de la bibliothèque WinCC (bibliothèque utilisateur, objets utilisateur, instruments à index) sont un exemple d'objets dynamiques.

L'objet prototype est ajouté comme copie du prototype dans la vue de process cible. Cette copie doit cependant être maintenant liée aux variables (de process) réelles de la gestion des variables.

Etape	Type	Configuration
5	Variable	Définissez pour la structure définie par l'opération 1 une variable (de process) devant être utilisée pour l'objet utilisateur.
6	Créer une instance	Créez une copie de l'objet prototype dans la vue de process par Glisser-Déposer à partir de la bibliothèque graphique. Liez cet objet à la variable (de process) avec l'assistant *Instancier le prototype* : L'assistant lie automatiquement tous les champs nécessaires de la variable structurée à la propriété correcte du composant de vue prototypique en remplaçant chaque liaison de variable prototypique, pour toutes les propriétés, par la liaison de variable concrète. Vous avez maintenant créé un objet qui pourra être rafraîchi au runtime avec les valeurs courantes des variables.

3.3.11.4 Instanciation dynamique

Outre l'assistant dynamique *Instanciation statique de prototype*, il existe également un assistant d'*instanciation dynamique de prototype*. Quelle est la différence par rapport à l'instanciation statique et quelles opérations doivent être modifiées ?
A la différence de la liaison statique d'un objet à des variables, les composants de vue peuvent être aussi liés dynamiquement. Cela signifie que l'instance ne sera activée au runtime qu'en fonction du contenu courant d'une variable. Par exemple, le composant décrit ci-dessus ne sera pas lié statiquement à la variable, mais le nom de la variable est dynamiquement géré. Le nom courant de la variable doit alors être déterminé via une variable de texte. Cette variable de texte, qui contient le nom courant d'une variable, doit être liée au composant de vue.
A la différence de l'instanciation statique, les opérations suivantes doivent être modifiées à la configuration:

Etape	Type	Configuration
2	Composant de vue	Configurez dans *Graphics Designer* un objet utilisateur ayant les propriétés définies par l'utilisateur comme décrit ci-dessus. L'objet utilisateur doit contenir un composant *texte statique* dont la propriété *texte* sera reprise comme composant défini par l'utilisateur. Cette propriété Texte reçoit le nom d'attribut *Tagname*. Ce nom de variable est utilisé pour la liaison dynamique des variables (de process).
5	Variable	Définissez pour la structure définie par l'opération 1 une variable (de process) devant être utilisée pour l'objet utilisateur.
6	Créer une instance	Créez une copie de l'objet prototype dans la vue de process par Glisser-Déposer à partir de la bibliothèque graphique. Liez cet objet à la variable (de process) avec l'assistant *Instancier le prototype*: L'assistant lie automatiquement tous les champs nécessaires de la variable structurée à la propriété correcte du composant de vue prototypique en remplaçant chaque liaison de variable prototypique, pour toutes les propriétés, par la liaison de variable concrète. Vous avez maintenant créé un objet qui pourra être rafraîchi au runtime avec les valeurs courantes des variables.

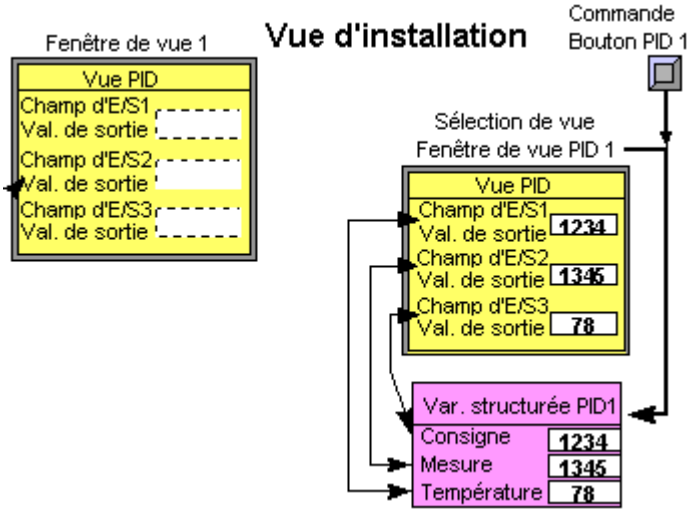
Lors de l'utilisation des objets utilisateur et des prototypes dynamisés, le concepteur s'assurera que les *actions C* nécessaires sont déjà mémorisées pour les objets. Celles-ci ne doivent pas être effacées, faute de quoi l'ensemble de la fonctionnalité du composant serait perdu.

3.3.11.5 Vues prototypes

La technique des vues prototypes va encore un peu plus loin. Le concept d'utilisation du prototype peut être organisé de manière si souple que toute modification d'un prototype se répercute automatiquement sur les objets créés. Cette souplesse nécessite des scripts C très flexibles qui doivent être liés.
La technique des vues prototypes utilise des vues 'modèle' pouvant être liées plusieurs fois à une ou plusieurs vues mère. Une vue modèle est simplement un 'patron' qui ne sera activé que dans un objet réel. La création d'un objet à partir d'un modèle (=vue prototype= patron) se fait par instanciation. Plusieurs instances (c.-à-d. objets réels concrets) peuvent être créées à partir d'un même modèle.

Modèle de fenêtre

Fenêtres utilisées avec données courantes du gestionnaire de données.



Les modifications ultérieures sont effectuées de manière centralisée (dans le modèle) et sont répercutées sur toutes les applications (instances). Ceci est une technique très efficace qui permet de faire l'économie des adaptations laborieuses sinon nécessaires dans toutes les applications.

Il est possible d'afficher jusqu'à 30 instances (objets) d'un type de modèle dans une image mère. Il est également possible d'utiliser plus de 30 objets en recourant à divers prototypes.

Les vues prototypes sont des composants de vue qui sont rangés après leur création dans la bibliothèque pour être réutilisés ensuite. Les composants de vue sont utilisés comme des instances de modèle dans les vues de process. Ces copies affichent les données courantes p. ex. de régulateurs ou de moteurs, qui sont visualisées dans le composant de vue. Les composants de régulateur ou de moteur correspondants sont affichés automatiquement.

Il est possible de créer des composants de vue simples ou complexes. Un composant de vue peut être composé de plusieurs éléments. Ceux-ci se recouvrent en tout ou partie mais forment une entité cohérente. Par exemple, toutes les données d'un moteur, telles que la vue de son état actuel, les données d'évolution du process, les données de maintenance, etc., sont regroupées dans un objet et rafraîchies en cas de besoin. Lorsque le process comporte de nombreux moteurs du même type, il suffit de faire des copies après avoir créé le composant de vue. Tout le reste est automatique.

La séquence ci-dessous est nécessaire pour la création:

Etape	Type	Configuration
1	Structures de données	Définissez, dans la gestion des variables, les structures à utliser dans le composant.
2	Structure de fenêtre	Configurez dans *Graphics Designer* le contenu du composant de vue, p. ex. bargraphes, champs d'E/S, etc.
3	Liez la structure et la construction de fenêtre	Liez les composants de vue aux champs de la structure à l'aide de scripts C spéciaux (tirés de l'exemple de projet).
4	Fenêtre de vue	Liez le composant de vue à l'objet de fenêtre de vue
5	Bibliothèque graphique	Placez la fenêtre dans la bibliothèque graphique

Le composant de vue peut être maintenant réutilisé dans les vues de process en le sélectionnant dans la bibliothèque graphique.
Examinons tout d'abord la création du modèle (Template).

Etape	néant	Configuration
1		Création d'une structure de variable (type de données type de structure) dans le gestionnaire de données ; vous définissez ici le nombre de variables élémentaires qui constitueront la structure (membres), leurs noms et leurs types de données respectifs (BIT, SHORT, WORD, TEXT, etc.). p. ex. PID avec valeur de consigne, valeur mesurée et température comme composants de structure.
2		Création d'une vue devant être utilisée comme modèle. Cette vue est typiquement plus petite que l'écran et peut être redimensionnée. Chaque vue créée peut être utilisée pour constituer un modèle (Template). La vue est créée à l'aide des fonctions graphiques d'édition ; des variables graphiques, telles que les champs E/S, les bargraphes, les indicateurs d'états, etc., y sont placées mais ne sont pas liées à des variables. Des relations internes (*liaisons directes*) entre objets graphiques, comme p. ex. le passage de la valeur de sortie d'un champ d'E/S à un bargraphe, déclenché par modification de valeurs de variables, sont configurées dans la vue.
3		Les champs graphiques sont maintenant liés aux champs de la variable structurée correspondante. Cette configuration de liaison ne crée des liens qu'au niveau du type (modèle uniquement) et pas encore avec des objets concrets du process. Il existe pour cela un exemple de projet préparé sur le CD WinCC (\Samples). La bibliothèque de projets (\Template) de ce projet contient un objet utilisateur *TemplateInit* qui effectue les mises en correspondance entre champs de variables structurées. Il se trouve dans la bibliothèque graphique et peut en être extrait et placé par 'Glisser-Déposer' dans une vue à concrétiser. L'objet *TemplateInit* possède déjà une logique de scripts toute prête. Celle-ci utilise une *Table de connexion* qui se remplit comme un tableau à la configuration et contient exactement les entrées susmentionnées et soulignées. La connexion entre les propriétés et les champs de la structure est ainsi effectuée. L'établissement de liaisons peut être activé à l'intérieur d'un

Etape	néant	Configuration
		modèle ou de l'extérieur. Il existe, dans la bibliothèque graphique de projets spécifique, des objets utilisateur qui ont l'aspect extérieur de boutons de commande, mais qui gèrent des informations paramétrables via le modèle (template) à invoquer. Tous les scripts servant à réaliser ces prototypes prêts à l'emploi doivent être chargés dans le projet cible. Voir à ce sujet les dernières opération 8-10 à la fin du chapitre. Les prototypes ne sont pas réalisables sans ces fonctions de projet.
4		Cette vue doit être utilisée comme boîte de process. Créez pour cela dans une vue temporaire (qui n'est nécessaire que pour cette étape intermédiaire) un objet fenêtre de vue et liez la propriété *Nom de vue* de cet objet à la vue contenant le composant de vue.
5		Placez cet objet de fenêtre de vue dans la bibliothèque graphique par un Glisser-déposer.

Ce modèle peut être maintenant réutilisé plusieurs fois dans les vues de process. La liaison aux variables de process s'effectue automatiquement par les noms attribués.

Etape	Type	Configuration
6	Définition de variables	Définissez une variable (de process) pour le type de donnnées associé (p. ex. PID_1 du type PID).
7	Créer une instance	Copiez le composant de fenêtre de vue à partir de la bibliothèque graphique Attribuez immédiatement le nom d'objet de la fenêtre de vue au nom de la variable (de process) devant être utilisée (p. ex. PID_1) : Sur PID_1, paramétrez *Objet de fenêtre de vue* → *Fenêtre de vue* →*Nom d'objet* →*Statique* GG

Lors de son placement, le composant de vue reçoit le nom d'une variable (de process) structurée dont les valeurs contiennent les données d'état d'un objet de process. Au runtime, le composant de vue se procure automatiquement les données d'état.

Ces vues prototypes utilisent certains scripts C qui sont ou ont été mémorisés comme fonctions de projet. Pour pouvoir utiliser les scripts C préparés, les scripts ci-dessous doivent être repris de l'exemple de projet. Procédez de la manière suivante:

Etape	Type	Configuration
8	Copie des fichiers de fonctions	Copiez à partir de l'exemple de projet, via <Projektpfad>\Library toutes les fonctions nécessaires (.fct), p. ex. LinkConnectionTable.fct dans votre répertoire de projet <Projektpfad>\Library.
9	Déclarer ces fonctions au projet	Appelez dans WinCC l'éditeur Global Script (*Global Script* → D → *Ouvrir*) pour déclarer ces nouvelles fonctions : Vous pouvez maintenant, à l'aide du bouton *Générer en-tête*, déclarer les nouvelles fonctions dans l'arborescence des fonctions du projet. Les nouvelles fonctions sont alors visibles dans la liste des fonctions de projet.
10	Reprendre des objets utilisateur	Les objets utilisateur sont disponibles dans la bibliothèque de projet. Dans un nouveau projet pour lequel vous n'avez pas encore mémorisé de symboles spécifiques dans la bibliothèque,

Etape	Type	Configuration
		vous pouvez simplement copier la bibliothèque depuis l'exemple de projet dans votre projet : Copiez <Projektpfad>\Library\library.pxl →dans votre chemin <Projektpfad>\Library! Sinon, vous pouvez transférer les objets utilisateur d'un projet à l'autre à l'aide du mécanisme d'exportation: Exportez les symboles souhaités à partir de l'exemple de projet comme fichiers .emf (*Fichier →Exporter)* et importez ces symboles .emf dans votre projet en sélectionnant *Ajouter → Importer*, dans une vue temporaire. Transférez les symboles dans votre bibliothèque de projet à l'aide de la fonction 'glisser-déposer'. Utilisez pour cela un dossier spécifique p. ex. 'Template'.

Les composants de vue (éléments de vue concrétisés ou modèles) permettent des économies de ressources importantes. Ils sont modélisés comme objets référencés à des variables et rangés p. ex. dans la bibliothèque graphique. Ils sont extraits de la bibliothèque graphique, placés dans la vue de process et rafraîchis automatiquement avec des données au runtime. La configuration de liaisons de variables à des éléments de graphisme tels que champs d'E/S, bargraphes, etc., n'est plus nécessaire.

Lorsque ces vues prototypes sont utilisées, il existe plusieurs manières d'affecter le nom courant à l'élément du composant de vue. Ceci s'effectue à l'aide des variables ci-dessous, que nous nommerons brièvement:

- Le nom d'instance est déterminé automatiquement à la **sélection** de la fenêtre de vue :
  pour cela une fonction de projet fournie par le système (EnabelTemplateInstance) est mémorisée dans la fenêtre de vue pour l'événement 'Sélection de vue'.
- Le nom d'instance est défini via une **variable E/S**, qui est lue automatiquement à l'aide d'un script dans la fenêtre de vue ; le nom d'instance est ainsi déterminé ;
  on utilise pour cela l'objet utilisateur préparé antérieurement *InstanceCallButton+Template*.
- Un **bouton de commande** passe le nom d'instance directement à la fenêtre de vue appelée :
  on utilise pour cela l'objet utilisateur préparé antérieurement *InstanceCallButtons+Template*.

Des exemples détaillés sont donnés pour ces variantes dans l'exemple de projet fourni sur le CD WinCC (\Samples).

3.3.11.6 Objets OCX

Les objets OCX et ActiveX sont des composants de vue disponibles sous forme de composants exécutables. Certains de ces composants (contrôles) sont proposés en standard par WinCC, p. ex. le Contrôle d'horloge numérique/analogique WinCC.

Ces composants peuvent être intégrés de manière simple aux vues de process.

Etape	Type	Configuration
1	Intégrer un objet OCX	Sélectionnez dans la palette d'objets de *Graphics Designer* le type *Contrôle OLE* (OCX) parmi les objets complexes. Ouvrez l'objet dans la vue de process avec , puis sélectionnez l'élément souhaité dans la boîte de dialogue affichée.
2	Lier propriétés	L'objet intégré possède également des propriétés et des événements. Les propriétés et événements disponibles pour un objet dépendent de l'OCX spécifique. Une liaison à une variable peut par exemple être réalisée avec la Propriété Couplage process.

Création

Les composants de vue OCX doivent être créés avec un environnement de développement spécial. Par exemple Visual C++ 5 de Microsoft ou Visual Basic 5 de Microsoft.
Les composants peuvent ainsi être modifiés ou affinés. Ils sont très performants, mais ne peuvent être créés avec un atelier de génie logiciel WinCC. Les composants OCX doivent toujours être créés ou modifiés en externe.
Par contre, la technologie des vues prototypes ne nécessite que des moyens propres à WinCC, contrairement à ce qui est le cas pour la programmation d'OCX performants et faciles à modifier. Il n'est alors pas nécessaire de posséder un savoir-faire en programmation d'OCX.

Un grand nombre de ces composants sont déjà disponibles aujourd'hui. Des 'Faceplates PCS7' prêts à l'emploi sont maintenant proposés dans le cadre de l'intégration des mondes du contrôle-commande et de la programmation d'automates programmables (PCS7) en contrôle de process.

Enregistrement

Les composants OCX créés ou achetés dans le commerce doivent être enregistrés sur la station WinCC. Il est possible de voir dans le dialogue de sélection, dans *Graphics Designer* (voir description ci-dessus), quels objets OCX sont disponibles sur une station WinCC. Tous les composants OCX enregistrés sur une machine sont listés dans le dialogue. Un composant OCX se mémorise sous forme d'un fichier ayant l'extension .OCX ou .dll.
Lorsqu'un composant n'a pas encore été enregistré, ceci peut être effectué dans le dialogue de contrôles OLE de WinCC. Cette boîte de dialogue contient un bouton de commande pour l'enregistrement et pour la suppression de l'enregistrement des composants courants sélectionnés. Pour la procédure d'enregistrement, le fichier correspondant doit se trouver sur la station WinCC.

La compatibilité et le bon fonctionnement des composants OCX doivent être testés par le concepteur. Seuls les composants OCX marqués WinCC ont été utilisés et testés dans l'environnement WinCC.

3.3.12 Configuration en ligne (runtime) - Remarques, restrictions

Un certain nombre de points sont à respecter pour la configuration en ligne.
Pour diverses raisons, quelques rares modifications ne peuvent pas être effectuées en ligne ou seulement dans certaines conditions ; il arrive également que les modifications ne deviennent actives qu'ultérieurement.

Control Center

Les modifications ci-dessous ne sont pas prises en compte

- Modification du type d'un ordinateur dans la liste d'ordinateurs

Les opérations de programmation ci-dessous ne sont pas possibles au runtime :

- Suppression/Renommage de variables
- Modification du type de données d'une variable

Alarm Logging

Les modifications ci-dessous ne sont pas prises en compte

- Modification des archives / journaux
- Modification des alarmes groupées
- Toute alarme contenant au total plus de 500 alarmes individuelles au runtime

Les opérations de programmation ci-dessous ne sont pas possibles au runtime :

- Aucune restriction

Tag Logging

Les modifications ci-dessous ne sont pas prises en compte

- Aucune restriction

Les opérations de programmation ci-dessous ne sont pas possibles au runtime :

- Les tableaux des archives utilisateur peuvent être créés mais pas modifiés.
- La suppression de données dans Tag Logging et dans les archives utilisateur.

Exception pour la configuration au runtime :

- Les tableaux des archives utilisateur peuvent être édités et supprimés via l'API de runtime de Tag Logging.

Global Script

Les modifications ci-dessous ne sont pas prises en compte

- La modification d'un script d'assistant n'est validée qu'au relancement de l'éditeur *Graphics Designer*.
- Scripts d'assistants modifiés.

Les opérations de programmation ci-dessous ne sont pas possibles au runtime :

- Aucune restriction

Report Designer

Les modifications ci-dessous ne sont pas prises en compte

- Modification du journal au fil de l'eau, étant donné que ce journal, une fois lancé, reste actif en permanence au runtime et ne recharge pas les informations de mise en page et de gabarit de ligne.

Les opérations de programmation ci-dessous ne sont pas possibles au runtime :

- Aucune restriction

Redundancy

Les modifications ci-dessous ne sont pas prises en compte

- Le nom d'ordinateur du partenaire ne peut être commuté sur un ordinateur tiers.
- L'aiguilleur AutoSwitcher ne peut être modifié : il faut configurer dès le début vers quelle cible doit se faire l'aiguillage. Il annule cependant l'aiguillage en cas de panne du partenaire.

Les opérations de programmation ci-dessous ne sont pas possibles au runtime :

- Aucune restriction

SIMATIC S7 Suite de protocoles ou canal S/7PMC

Les modifications ci-dessous ne sont pas prises en compte

- Les paramètres de diagnostic via S7Chn.ini (pas édité) ne sont pas pris compte en mode en ligne.
- Toutes les modifications d'adresses de communication sont certes prises en compte en ligne mais ne sont exploitées que lors de l'établissement d'une liaison.

Les opérations de programmation ci-dessous ne sont pas possibles au runtime :

- Aucune restriction

Text Library

Les modifications ci-dessous ne sont pas prises en compte

- Aucune restriction
  - Dans le module Text Library, les textes modifiés sont pris en compte par sélection de *Fichier → Transmettre modifications au projet actif*
  - Dans Alarm Logging, les modifications sont prises en compte dans la bibliothèque de textes via *Fichier →Enregistrer*.

Les opérations de programmation ci-dessous ne sont pas possibles au runtime :

- Aucune restriction

User Administrator

Les modifications ci-dessous ne sont pas prises en compte

- Les modifications des droits d'accès des utilisateurs ne sont activées qu'après une nouvelle connexion ou déconnexion.

Les opérations de programmation ci-dessous ne sont pas possibles au runtime :

- Aucune restriction

4 Cours de programmation en C - WinCC

Le langage de programmation de script WinCC s'avère, comme le montre les expériences faites jusqu'ici, une tâche difficile à résoudre pour les non initiés de la programmation en langage C. Pour de nombreuses applications, il suffit cependant de savoir paramétrer et utiliser les fonctions existantes.

Nota :
Il n'est pas absolument nécessaire de créer des fonctions ou actions pour dynamiser les objets, un *Dialogue dynamique* étant disponible pour l'activation. Ce dialogue crée des *actions C* exécutables et extensibles. Les *dialogues dynamiques* sont exécutées dans une zone spéciale de la mémoire du système d'exploitation et sont par conséquent nettement plus rapides que les *actions C*. Les fonctionnalités des *dialogues dynamiques* sont cependant limitées. Un *dialogue dynamique* peut être ultérieurement transformée en une *action C* mais avec une perte de performance.
La méthode de dynamisation la plus rapide est cependant la *connexion directe*. Mais il existe ici aussi quelques restrictions par rapport aux *actions C* (p. ex. uniquement pour les objets dans la vue).

Les utilisateurs devant maîtriser des tâches très exigeantes disposent avec les *actions C* de *Graphics Designer* et avec l'éditeur *Global Script* d'un langage de script très puissant. Lorsque, dans l'euphorie initiale de développement, les scripts deviennent longs et complexes, l'utilisateur peut les intégrer à des fonctions exécutables *dll (Dynamic LinkLibrary)* dans WinCC avec un gain de performance net.

Groupe cible

Ce cours de programmation en langage C est destiné à initier les novices aux éléments essentiels du langage C. Pour ceux qui programment déjà en C, ce cours leur présentera les particularités de l'environnement de développement.

Les exemples utilisés en ce cours ont été configurés dans le projet WinCC *Cours_00*.

Vue d'accueil du cours de C

Nous avons pour chaque thème de ce cours configuré une vue présentant plusieurs exemples. Les différentes vues sont accessibles par sélection du bouton correspondant dans la barre de touches.

Le bouton *Diagnostic* permet d'activer/de désactiver la fenêtre de diagnostic WinCC. Dans les *actions C*, toutes les sorties déclenchées par *printf* sont affichées dans cette fenêtre de diagnostic configurée.
La sélection du bouton correspondant aux divers exemples avec **D** permet d'afficher le code source correspondant à l'exemple sélectionné.

4.1 Environnement de développement de scripts dans WinCC

WinCC met avec ses *Actions C* dans *Graphics Designer* et avec l'éditeur *Global Script*, un langage de script puissant à la disposition de l'utilisateur. La syntaxe de ce langage de script est conforme au standard du langage C (selon normalisation ANSI) et est interprétée. Cela signifie que les instructions formulées dans le langage de programmation de script font uniquement l'objet d'un contrôle de syntaxe à la traduction et qu'elles ne sont pas exécutées immédiatement.

Global Script est le terme générique désignant les *fonctions C (fonctions standard et de projet)*et les *actions globales*, qui selon leur type, peuvent être utilisées dans un ou plusieurs projets.

4.1.1 Fonctions et actions dans WinCC

Les *actions C* (fonctions et actions) sont exécutées au runtime et servent à visualiser et à conduire le process.
Les fonctions et actions permettent de modifier les propriétés d'objet et de réagir à des événements.

Structure d'une fonction

Les *fonctions* sont composées de deux zones :

En-tête de fonction	`Type de la valeur retournée`
	`Nom de fonction`
	`(Type du paramètre1 Nom de paramètre1,`
	`....)`

Corps de fonction	`{`
	`déclarer variables ;`
	
	`commandes ;`
	
	`return (valeur retournée);`
	`} // Fin de fonction`

En-tête de fonction

L'en-tête décrit l'environnement de la fonction (interface vers l'extérieur).
Exemple d'*en-tête de fonction*:

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
```

L'en-tête de la *fonction* décrit le nom (comment la fonction est-elle appelée ?) et les paramètres de transfert (quelles données sont passées à la *fonction* et quelle *valeur de retour* est délivrée par la fonction ?) de la fonction.

L'*en-tête de la fonction* est affichée automatiquement à l'appel de l'*action C* ; il n'est pas nécessaire de la configurer.
L'*en-tête de la fonction* indique à la personne effectuant la configuration la *valeur retournée* qui sera passée et quelles données sont déjà disponibles dans la fonction.

Les en-têtes de fonction affichées sont différentes selon qu'il s'agit de la dynamisation d'une propriété ou d'un événement.

En-tête de fonction pour événements

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
                   lpszPropertyName)
{
//Corps de fonction
}
```

La *valeur retournée* est pour ces actions toujours du type *void* (rien). La commande *return;* peut ne pas figurer dans les *actions C*, cette commande étant toujours exécutée automatiquement à la fin d'une *action C*.

En-tête de fonction pour une propriété

```
#include "apdefap.h"
long _main(char* lpszPictureName, char* lpszObjectName, char*
                  lpszPropertyName)
{
//Corps de fonction avec valeur retournée
}
```

Pour les *actions C* de propriétés d'objet, le nouveau paramétrage de la propriété à dynamiser est passé par la *valeur retournée* actuelle de l'action C. Le type de la *valeur retournée* est défini dans l'en-tête de l'*action C*, par exemple un nombre (dans notre exemple type *long*) ou une chaîne de caractères (p. ex. *char* *) :

Corps de fonction

Le corps de la fonction décrit le contenu de la fonction (**que fait** la fonction ?)

```
/* Déclaration des variables */
intnumber;

/* Affectation de valeurs aux variables */
number=123;

/* Affichage des valeurs de variables dans la fenêtre de diagnostic */
printf("Le premier chiffre est %d\n",number1);
}
```

Le *corps de fonction* contient les différentes instructions. Le *corps de fonction* est annoncé par une accolade ouvrante. L'accolade fermante marque la fin de la fonction.
Certaines données complémentaires nécessaires pendant l'exécution de la *fonction* (données locales) sont définies dans le *corps de fonction*. A la fin de l'exécution de la *fonction*, ces données sont automatiquement effacées.
Les instructions contenues dans le *corps de fonction* sont terminées par un point-virgule.

Valeur retournée

La *valeur retournée* de la *fonction* (par exemple la nouvelle couleur ou une nouvelle position X) est restituée à la fin de la fonction dans le *corps de fonction*.

La formulation de séquences en C fait appel aux fonctions déjà existantes. Ces fonctions peuvent être affichées et utilisées à l'aide de l'arbre de fonctions. Ces fonctions spécifiques à un projet ou utilisables dans tous les projets sont formulées dans l'éditeur *Global Script*. On distingue les types suivants :

Fonctions standard

Vous pouvez créer de nouvelles *fonctions standard* et modifier les fonctions existantes.
Les *fonctions standard* sont connues de tous les projets et placées par catégorie.
Ces fonctions sont exécutées dans *Control Center* avec l'éditeur *Global Script.*

Les *fonctions standard* servent à dynamiser les *objets graphiques* et les *archives.* Les *fonctions standard* sont appelées dans des *fonctions de projet*, dans d'autres *fonctions standard* et dans les *actions* .

Fonctions de projet

Vous pouvez créer de nouvelles *fonctions de projet* et modifier les fonctions existantes. Les *fonctions de projet* ne sont connues que du *projet* dans lequel elles ont été créées. Ceci constitue la **seule différence** par rapport aux *fonctions standard.*
Ces fonctions sont exécutées dans *Control Center* avec l'éditeur *Global Script.*

Les actions utilisées plus d'une fois doivent être définies comme *fonctions de projet* ou comme *fonctions standard*.
Les *fonctions de projet* peuvent et doivent être catégorisées comme les *fonctions standard* et les *fonctions internes*.
Les avantages des *fonctions de projet* sont les suivants :

- leur **réutilisabilité.** Lorsqu'une *action C* a été configurée et suffisamment testée, elle peut être réutilisée à tout moment sans configuration complémentaire et sans subir de nouveaux tests importants.
- facilité de **modification**. Lorsqu'une *action C* a subi une modification, cette modification n'est exécutée qu'en un seul endroit – dans la *fonction projet*. Il n'est plus nécessaire de rechercher et d'adapter les dynamisations configurées dans toutes les vues. Cela facilite non seulement la configuration mais aussi la maintenance et le diagnostic d'erreurs.
- **volume d'image réduit** et par conséquent formation plus rapide de l'image. Lorsque des fonctions de projets ou des fonctions standard sont utilisées dans des actions concernant des vues, le volume d'éléments d'image est réduit.
- Les fonctions de projet peuvent être protégées contre les modification par un mot de passe (**Autorisation d'accès**) (Protection des données de configuration et du savoir-faire de configuration)

Fonctions internes

Les *fonctions internes* ne peuvent être ni créées ni modifiées et sont connues de tous les projets
Les *fonctions internes* s'utilisent pour dynamiser des *objets graphiques*, *archives*, dans des *fonctions de projet*, dans d'autres *fonctions standard* et dans les *actions*.
Les *fonctions internes* se divisent en catégorie.

Actions

Les actions (globales et sur l'objet) peuvent être créées et modifiées. Ces actions ne sont connues que du projet dans lequel elles ont été créées.
Les fonctions de projet, standard et internes s'utilisent dans les cas suivants :

- dans les *actions C* liées à des objets.
- dans les *actions* que vous créez dans le *dialogue dynamique*.
- pour dynamiser des *alarmes*, *archives de valeurs de process*, *archives condensées* et *archives utilisateurs*.

Nota :
Ces actions sont interprétées. Il faut donc s'attendre à une forte charge du système en cas d'utilisation d'actions nombreuses ou volumineuses. Il est par conséquent préférable de remplacer les actions volumineuses par des *DLL* (Dynamic Link Libraries) mieux adaptés.

Les actions s'utilisent au runtime pour la conduite du process. Leur exécution est déclenchée par un *Déclencheur* (événement déclencheur). Il faut pour cela activer le projet.

Une action globale se créent dans l'éditeur *Global Script*. Lancez cet éditeur dans *Control Center*.

4.1.2 Les éditeurs de fonctions et d'actions

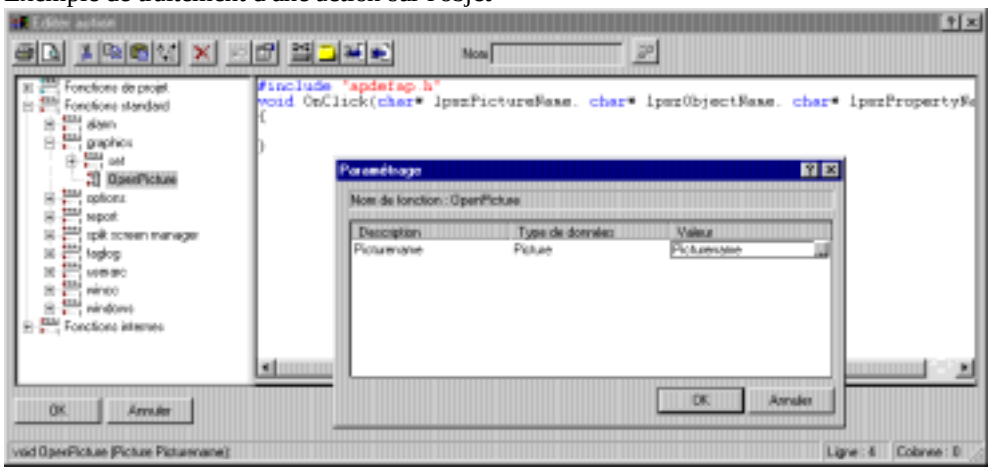
Les fonctions et actions globales sont traitées dans l'éditeur *Global Script.*
Les fonctions se subdivisent en fonctions de projet et fonctions standard. Ces fonctions sont utilisées dans des vues sous forme *d'action C* appliquée à l'objet ou dans le système d'alarme.
Les actions globales par contre sont indépendantes de tout objet et ne sont exécutées que lorsqu'interviennent certains événements (déclenchement par variable ou par un temps).

Ces deux éditeurs se distinguent par le mode d'appel et l'étendue de leur fonctionnalité.

L'éditeur d'actions sur des objets

Cet éditeur s'utilise dans *Graphics Designer* pour la dynamisation des *objets*. Cet éditeur ne permet de traiter que le *corps de fonction*. L'*en-tête de fonction* est créé automatiquement à l'appel.
Le dialogue de la vue ci-dessous permet le paramétrage de la fonction introduite.

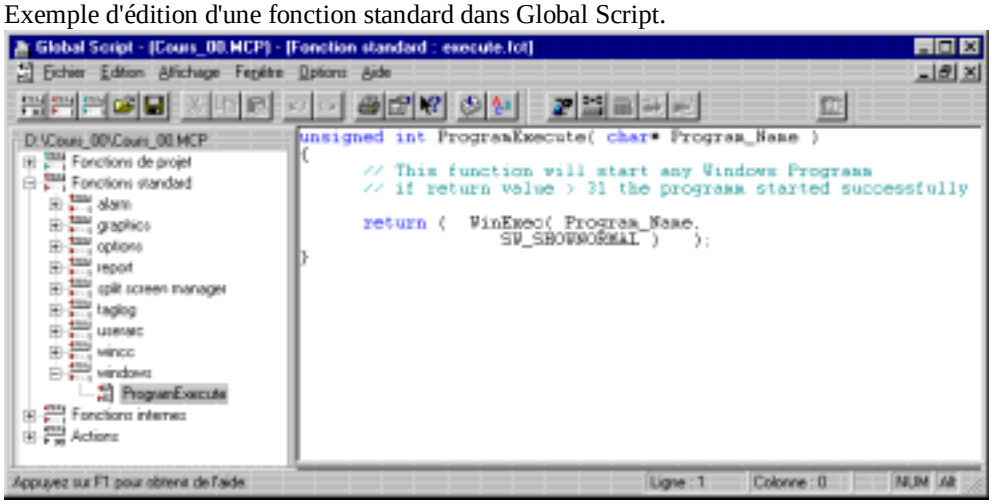
Exemple de traitement d'une action sur l'objet



L'éditeur Global Script

L'éditeur est appelé dans *Control Center*. Il permet d'éditer des *fonctions* et des *actions*. Il ne permet cependant pas de traiter les *actions* sur l'*objet*. L'*en-tête de fonction* est protégée pendant l'exécution des *actions*, c'est-à-dire qu'elle ne peut être modifiée. Pour les *fonctions*, l'*en-tête de fonction* et de *corps de fonction* sont éditables.

Exemple d'édition d'une fonction standard dans Global Script.



Les fonctions de manipulation ci-dessous sont valables pour les deux éditeurs :

Touches de commande dans l'éditeur

Fonction	Touche
Nouvelle ligne	Valeur retournée
Effacement caractères vers la droite	SUPPR
Effacement caractères vers la gauche	Touche de retour
Saut en début de ligne	POS1
Saut en fin de ligne	FIN
Saut au début du texte	Ctrl+POS1
Saut en la fin du texte	Ctrl+FIN
Déplacement de la marque d'insertion	avec les touches de curseur
Couper texte marqué	Ctrl+X
Copier texte marqué	Ctrl+C
Insérer texte à partir du presse-papier	Ctrl+V

Commandes de la souris dans l'éditeur

Fonction	Commande souris
Marquer texte	avec bouton gauche
Marquer un mot	double clic avec bouton gauche
Déplacer marque d'insertion	avec bouton gauche

Autres fonctions d'édition :

Fonction	Signification
Mode d'écriture	Insertion
Texte marqué	Ce texte sera remplacé par le prochain caractère entré au clavier.
Marquer une zone de texte	Placer la marque d'insertion au début du passage à marquer en maintenant enfoncée la touche MAJ ; placez la marque d'insertion à la fin du passage marqué
Extension d'un marquage	Maintenir la touche MAJ enfoncée ; placez la marque d'insertion à la fin du passage marqué

4.1.3 Création de fonctions et d'actions

Création d'une nouvelle fonction

Les opérations ci-dessous sont nécessaires à la création d'une fonction standard ou d'une fonction de projet :

Opération	Procédure: Création d'une fonction standard ou de projet
1	Formuler la fonction
2	Compléter les informations de fonction
3	Traduire la fonction
4	Mémoriser et, le cas échéant, renommer la fonction
5	Le cas échéant générer les fichiers d'en-tête

Vous avez ainsi créé votre fonction standard ou de projet.

Le fichier d'en-tête *apdefap.h* sert à lier le fichier d'en-tête *ap_glob.h*, qui contient la déclaration des *fonctions standard*.
Les fichiers *apdefap.h* et *ap_glob.h* se trouvent dans le répertoire *<Répertoire d'installation WinCC>\APLIB*.
Lorsqu'une nouvelle *fonction standard* est créée et qu'elle utilise d'autres *fonctions standard*, le fichier d'en-tête *apdefap.h* ou *ap_glob.h* doit être lié.

Les nouvelles *fonctions standard* créées sont mémorisées avec les *fonctions standard* existantes sur l'*ordinateur serveur* dans le répertoire *<Répertoire d'installation WinCC> \APLIB*.
Les *fonctions de projet* définies sont par contre mémorisées sur l'*ordinateur serveur* dans le répertoire *<Nom projet>\Library* . Les en-têtes de fonction (appelées prototypes de fonction) des nouvelles *fonctions de projet* sont rangées dans le fichier de déclaration (pour fonctions C) *Ap_pbib.h* dans le répertoire*<Nom projet>\Library*.

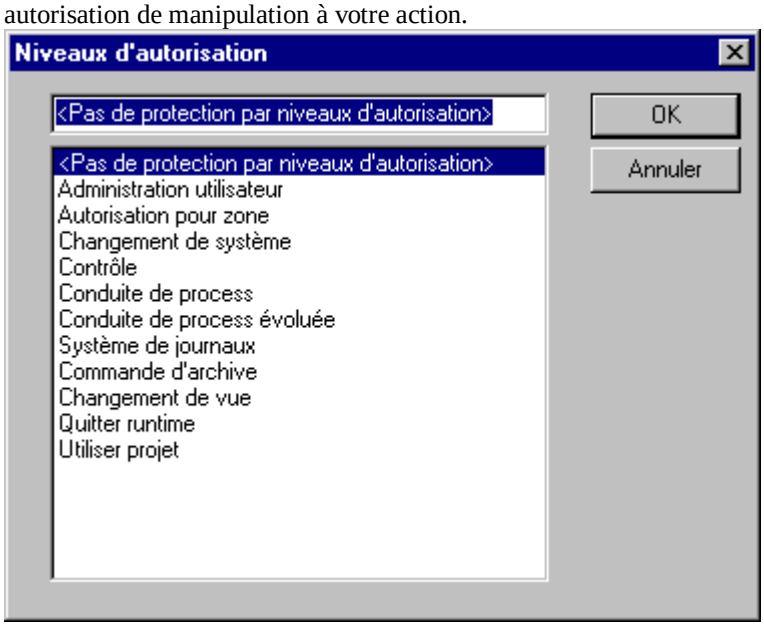
Création d'une nouvelle action

Les opérations ci-dessous sont nécessaires à la création d'une action globale:

Opération	Procédure: Création d'une action
1	Formuler l'action
2	Compléter les informations d'action
3	Définir le ou les déclencheurs devant constituer l'événement déclencheur
4	Traduire l'action
5	Régler l'autorisation de manipulation
6	Mémoriser et, le cas échéant, renommer l'action

Vous avez ainsi créé une action globale.

Dans *Global Script* , vous avez la possibilité de lier, avec *Edition →Autorisation d'accès* , une autorisation de manipulation à votre action.



Le fichier d'en-tête *apdefap.h* se trouvant dans le répertoire de projet *...\<Nom projet>\LIBRARY'* est lié automatiquement lors de la création d'une nouvelle action.

En l'absence d'une *fonction de projet*, le fichier d'en-tête rangé dans le répertoire *...\APLIB* est lié. Les fonctions *standard* et de *projet* sont ainsi disponible dans les actions.

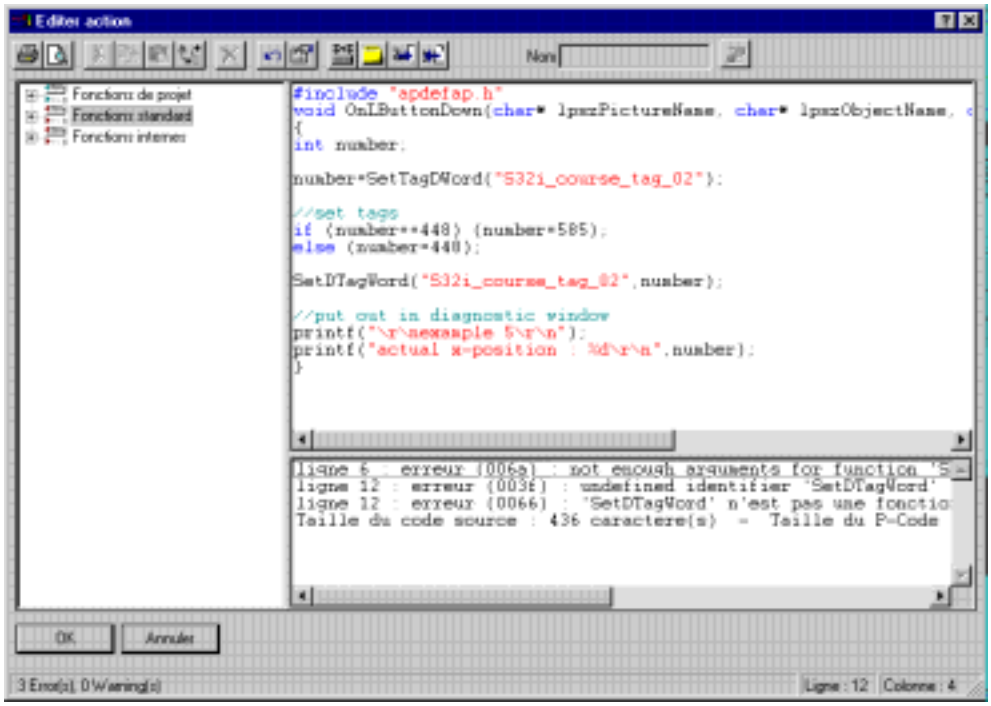
On opère une distinction sur les systèmes multipostes entre actions globales et actions locales.

Les actions globales sont rangées indépendamment des ordinateurs dans le répertoire des projets (*...\<Nom projet>\PAS*). Ces actions sont actives sur tous les ordinateurs.
Les actions locales sont mémorisées sur un ordinateur auquel elles sont liées dans le répertoire de projet (*...\<Nom projet>\<Nom ordinateur>\PAS*). Les actions locales ne sont par conséquent actives que sur l'ordinateur sur lequel elles se trouvent.

Toutes les actions globales définies pour un projet sont activées si le module runtime *Global Scripts* figure dans la liste de démarrage. L'activation intervient au moment du passage au mode runtime.

4.1.4 Test des fonctions et actions

Messages d'erreur



Les messages de syntaxe (*Errors* ou *Warnings*) affichés dans la *fenêtre d'erreurs* se trouvent dans n'importe quel manuel de programmation en C ou manuel de compilateur C (p. ex. Visual C de Microsoft). Il s'agit de messages d'erreur standard émis par n'importe quel compilateur C lors du contrôle de la syntaxe.
Les *Errors* ont toujours pour conséquence que l'action n'est pas exécutable, alors que les *Warnings* ne sont que des avertissements signalant d'éventuelles erreurs de traitement.

Nota :
Les avertissements affichés dans la fenêtre d'erreur doivent être dans tous les cas pris en compte. Ils signifient qu'une correction est nécessaire pour garantir une exécution correcte du programme. Les avertissements peuvent concerner par exemple des formats de nombre erronés lors des appels de fonction. L'élimination des avertissements dénote par ailleurs un style de programmation élégant !

Elimination des erreurs de syntaxe

La ligne d'erreur affichée dans la *fenêtre d'erreur* de l'éditeur d'actions C décrit l'erreur de syntaxe en détail en indiquant la ligne de code. Un **GG** sur la ligne d'erreur place automatiquement le curseur dans la ligne de code correspondante. L'erreur peut ainsi y être directement analysée et rectifiée.

Sorties des tests des actions C

L'exécution d'une *action C* peut être testée à l'aide d'instructions de sortie propres. Ces instructions de sortie doivent être écrites avec la fonction de sortie de C (*printf*).
Les instructions de sortie sont sorties dans la fenêtre de diagnostic lorsque l'exécution de l'action est terminée. La fenêtre de diagnostic se configure dans la vue sous *Graphics Designer* à l'aide d'*objets complexes* →*Fenêtre d'application* →*Global-Script* →*Diagnostic_GSC*.

4.1.5 Importation et exportation de fonctions et actions

Les actions globales peuvent être exportées et importées.
Une action importée remplace complètement l'action se trouvant dans la fenêtre active.

Les *actions C* déjà formulées peuvent être réutilisées de différente manière :

Rangement d'une action C dans le presse-papier

Lorsque l'*action C* formulée doit être réutilisée immédiatement dans le masque d'action suivant (d'une autre propriété ou d'un autre événement), l'*action C* actuelle ou des extraits de cette *action C* peuvent être exportés ou importés via le presse-papier. Le *bouton Copier/Insérer* de la barre d'outils ou les raccourcis clavier (*CTRL C, CTRL V*) permettent ces exportations ou importations.

Exportation d'actions C

Une *action C* écrite peut être rangée à tout moment dans un fichier, par exemple *Nom vue.act* également pour des opérations de copie (copie d'actions pour d'autres propriétés ou événements. Sélectionnez pour cela le bouton *Exporter action* et attribuez un nom de fichier. Ceci permet de ranger ou de mettre à disposition les actions basiques fréquemment utilisées à tout moment dans le chemin (*<Nom projet>\GraCS\)*.

Importation d'actions C

Une action C existante mise à disposition dans le chemin de vue de projet (*<Nom projet>\GraCS*) peut être à tout moment insérée et modifiée dans le masque d'action. Sélectionnez la *Propriété* à dynamiser et appelez l'éditeur d'*actions C*. Sélectionnez le bouton *Importer action* et sélectionnez le nom de fichier contenant l'action souhaitée, p. ex. *Nom vue.act*. Ceci permet d'utiliser à tout moment et d'adapter ultérieurement les actions de base à la dynamisation.

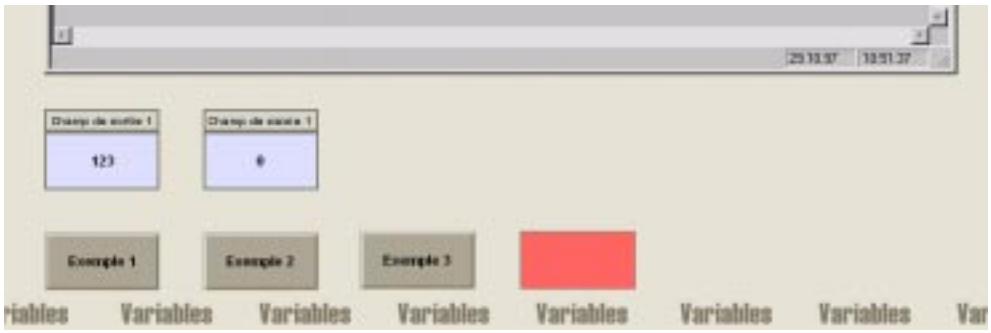
Rangement d'un objet configuré et lié comme objet de bibliothèque

Les objets déjà préconfigurés avec des *actions C* complexes peuvent être rangés comme objet graphique global avec dynamique dans la *bibliothèque de projets* pour leur utilisation ultérieure. Il suffit pour cela de copier l'objet par *glisser-déposer* (sélection et copie avec la souris) dans la *bibliothèque de projets* . Les actions configurées pour les *propriétés* ou *événements* sont en outre reprises. Un exemple d'objet déjà mémorisé avec des *actions C* figure dans le dossier Objets utilisateurs

4.2 Variables en C

Sélectionnez le bouton *Variables* pour accéder aux exemples concernant les variables dans notre projet. Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.

Vue de variables



Un clic de la souris sur le bouton *Variable* permet d'afficher la vue ci-dessus. Les différents boutons de la figure correspondent aux exemples décrits. Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.

Un peu de théorie sur les variables avant d'examiner nos exemples.
Les variables sont constituées d'un *nom de variable*, d'un *type de variable* et d'un contenu. Les variables peuvent être comparées à des *contenants* . Nous donnons à ce *contenant* un nom unique de variable, car nous voulons pouvoir plus tard localiser et utiliser ce *contenant* et son contenu.

Les variables C ne doivent pas être confondues avec les variables WinCC. Les variables ici décrites ne sont disponibles que dans les *Scripts*.

Types de variable en C

Le langage C connaît les types de base de variable correspondant aux intervalles de valeurs ci-dessous :

Type	Intervalle de valeur
int	Nombres entiers (nombres sans chiffre derrière la virgule) p. ex. 2 / -3 / 5 / etc.
char	Un caractère p. ex. a / k / z , mais aussi la touche RETOUR
float	Nombres à virgule flottante (nombres avec des chiffres derrière la virgule) p. ex. 3.23 / 4.32 / 3.01
double	Nombres à virgule flottante avec intervalle de valeur double de celui du type Float

Le type *double* ne se distingue du type *float* que par l'intervalle de valeur. Le type *double* permet une représentation plus précise des nombres (plus grande résolution).

Le type de variable *int* peut être précédé des mots clé *signed* (signé) ou *unsigned* (non signé). Lorsqu'il est précédé de *signed*, le programme interprète le signe ; lorsque les nombres sont précédés de *unsigned*, seules les valeurs positives peuvent être utilisées.
Le type *int* peut également être précédé de *long* ou *short*.

Sauf convention contraire explicite, toutes les variables *int* sont du type *signed int*.

Intervalle de valeur des types de variable :

Type	Intervalle de valeur
int	- 2 147 483 648 à 2 147 483 647
unsigned int dans WinCC également (**DWORD**)	0 à 4 294 967 295
short int dans WinCC également (**SHORT**)	- 32 768 à 32 767
long int	- 2 147 483 648 à 2 147 483 647
unsigned short int	0 à 65 535
unsigned long int	0 à 4 294 967 295
char	tous les signes ASCII
unsigned char	tous les signes ASCII
float	-10 ^38 à 10 ^38
double	-10 ^308 à 10 ^308

Il y avait autrefois deux types de caractères ASCII, (il s'agit de caractères pouvant être représentés), à coder. Le premier intervalle de valeur s'étendait de -128 à 127 et le second de 0 à 255.
Pour palier le problème des différentes valeurs ASCII, le type *unsigned char* (caractère non signé) a été introduit. Ce type ne connaît que les valeurs de 0 à 255.
Seul le type *unsigned char* est encore utilisé aujourd'hui.

Paramètre pour printf

Dans nos exemples, les résultats sont sortis avec *printf* dans la fenêtre de diagnostic de *Global Script*. Ci-après quelques informations sur *printf*.
Structure de *printf* et exemple :

```
printf("Contenu de la variable: %d\r\n",variable);
```

Le caractère *%* est un caractère spécial comme le caractère \. Le *%* indique au programme que la valeur d'une variable doit figurer à cet endroit. *%d* indique une *variable type integer* (entier). Dans les énumérations de variables, les variables sont séparées par une virgule.

Ci-dessous un court extrait des paramètres de *printf:*

Paramètre	Signification
%d	int, short int ou char (comme nombre décimal)
%ld	représentation d'un nombre du type long int
%c	comme %d avec char comme caractère
%x	comme %d avec char comme nombre hexadécimal avec petit a...f
%X	comme %x mais avec nombres hexadécimaux avec grand A...F
%o	comme %d avec char comme caractère octal
%u	comme %d mais uniquement valeurs non signées
%f	Nombres du type float, notation virgule flottante, p. ex. 3.43234
%e	Nombres du type float, notation exponentielle, p. ex. 23e+432
%E	comme %e mais avec E majuscule, p. ex. 23E+432
%g	Nombres du type float, notation exponentielle ou virgule flottante
%s	char* ou char[]
%le	Présentation double nombre
%%	N'est pas une instruction mais sort un %
\n	N'est pas une instruction mais sort un retour chariot (nouvelle ligne)
\r	N'est pas une instruction mais sort un Line feed (saut de ligne)
\t	N'est pas une instruction mais sort un tabulateur

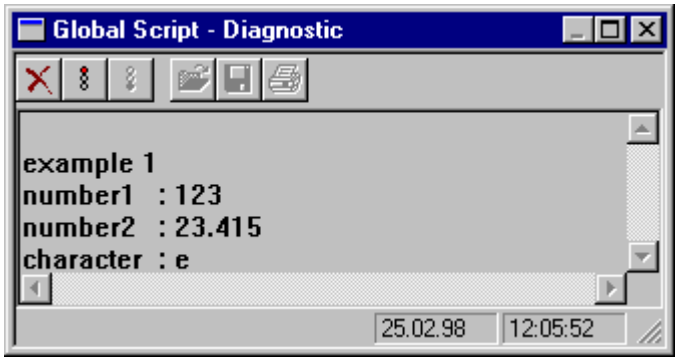
4.2.1 Exemple 1 - Utilisation de types de variable

Nous vous allons montrer dans notre exemple comment on utilise les différents types de variable dans une *action C*.
L'action est configurée sur le bouton *exemple_01* → *Evénement* →*Souris* → *Cliquez bouton gauche*.
Lorsque le bouton *Exemple 1* est cliqué avec la souris, le script ci-dessous est édité.

Action C-

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//tag déclaration
intnumber;
float number2 = 23.415;//set value in declaration
char  character;

//set tag values
  number1  =123;
character = 'e';

//output in diagnostics window
printf("\r\nexample 1\r\n");
printf("number1\t : %d\r\n",number1);
printf("number2\t : %.3f\r\n",number2);
printf("character : \t%c\r\n",character);

//set internal tag
SetTagDWord("S32i_course_tag_03",number1);
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- Les variables (*numéro1*, *numéro2* et *caractère*) sont créées dans la section *Déclaration* . Ceci s'effectue par la notation *Type de variable Nom de variable* et chaque ligne est terminée par un *;*.
- Pour la variable *numéro2* , l'affectation de valeur s'effectue avec la déclaration. La notation est *Type de variable Nom variable = Valeur* à affecter.
- Des valeurs sont affectées aux variables restantes. Nous avons maintenant des *variables* d'une taille définie (*int*, *float* et *char*).
- Les *variables* ont maintenant les valeurs (*123*, *23.415*, et *e*). Les chiffres décimaux sont séparés du reste des nombres par un point.
- Les valeurs sont alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.

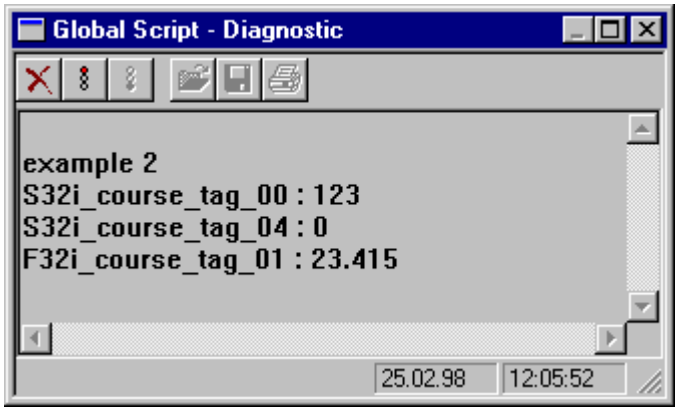
4.2.2 Exemple 2 - Variable C en liaison avec des variables de WinCC

Nous montrerons dans notre deuxième exemple l'utilisation de variables C en liaison avec des variables WinCC appelées ici *variables internes*. Les *variables externes* (variables de process de la gestion des variables) sont lues et modifiées de la même manière.
Cet exemple présente l'affectation à une variable (interne ou externe) d'une valeur définie *(SetTag)* ainsi que la lecture de la valeur actuelle d'une variable *GetTag)* .
L'action est configurée sur le bouton *exemple_02* → *Evénement* →*Souris* →*Cliquez bouton gauche* .
Lorsque le bouton *Exemple 2* est cliqué avec la souris, le script ci-dessous est édité.

Action C

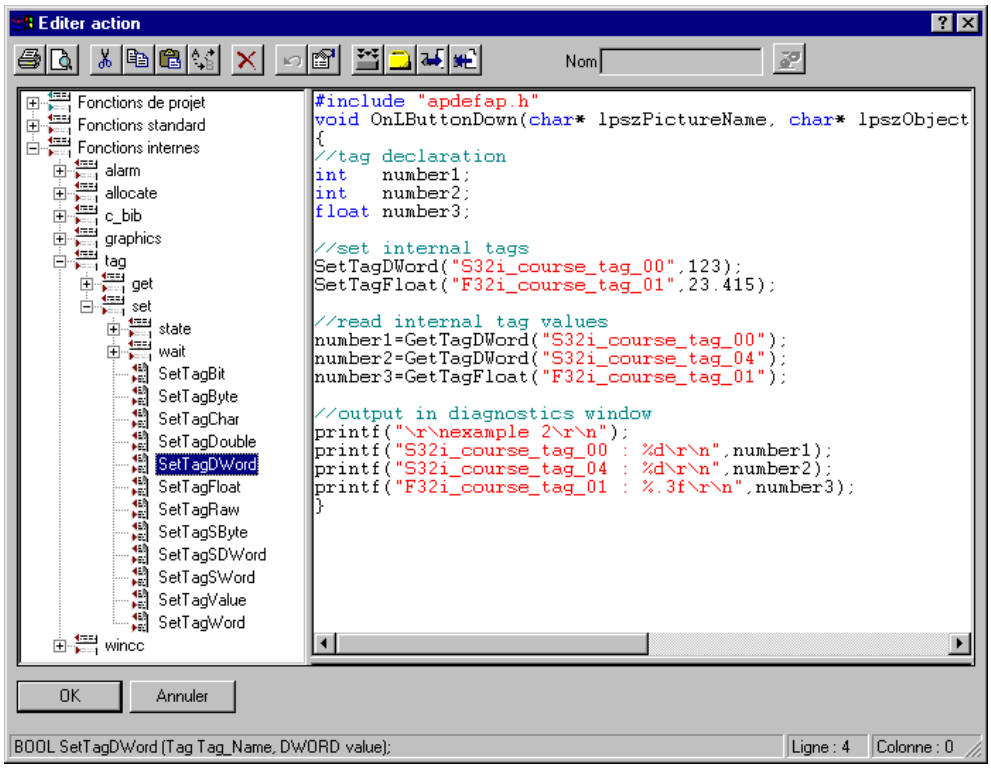
```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//tag déclaration
intnumber;
int  number2;
float number3;

//set internal tags
SetTagDWord("S32i_course_tag_00",123);
SetTagFloat("F32i_course_tag_01",23.415);

//read internal tag values
number1=GetTagDWord("S32i_course_tag_00");
number2=GetTagDWord("S32i_course_tag_04");
number3=GetTagFloat("F32i_course_tag_01");

//output in diagnostics window
printf("\r\nexample 2\r\n");
printf("S32i_course_tag_00 : %d\r\n",number1);
printf("S32i_course_tag_04 : %d\r\n",number2);
printf("F32i_course_tag_01 : %.3f\r\n",number3);
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C.
- Dans la section *Set internal tags* , des valeurs (*123* et *23.415*) sont affectées aux variables internes WinCC (*S32_course_tag_00* et *F32i_course_tag_01*) à l'aide des fonctions WinCC prévues à cet effet *SetTagDWord* et *GetTagFloat*.



- Dans la section suivante, nous lisons à l'aide de la fonction WinCC correspondante (*GetTagDWord, GetTagFloat*) la valeur des variables WinCC internes (*S32i_course_tag_00, S32i_course_tag_01*) et nous l'affectons aux variables C (*numéro1, numéro3*).
- La valeur de la variable (*S32i_course_tag_04*) est lue dans le champ de saisie 1 et affectée à la variable C (*number2)* . Cette valeur peut être modifiée. Cliquez avec la souris dans *champ de saisie 1*, introduisez la valeur et terminez l'introduction avec *Return*.
- Les valeurs sont alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.

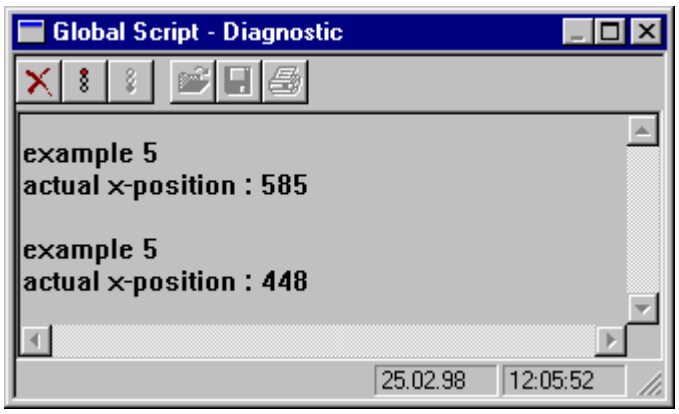
4.2.3 Exemple 3 - Utilisation de variables

Notre troisième exemple présente une autre possibilité d'utiliser une variable. Le contenu de la variable est changé en cliquant avec la souris. Cette variable est liée à la position X d'un objet. L'objet est déplacé lorsque la variable est modifiée.
L'action est configurée sur le bouton *exemple_03* → *Evénement* →*Souris* →*Cliquez bouton gauche* .
Lorsque le bouton *Exemple 3* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
int number;

number=GetTagDWord("S32i_course_tag_02");

//set tags
if (number==448) (number=585);
else (number=448);

SetTagDWord("S32i_course_tag_02",number);

//output in diagnostics window
printf("\r\nexample 5\r\n");
printf("actual x-position : %d\r\n",number);
}
```

Affichage dans la fenêtre de diagnostic :



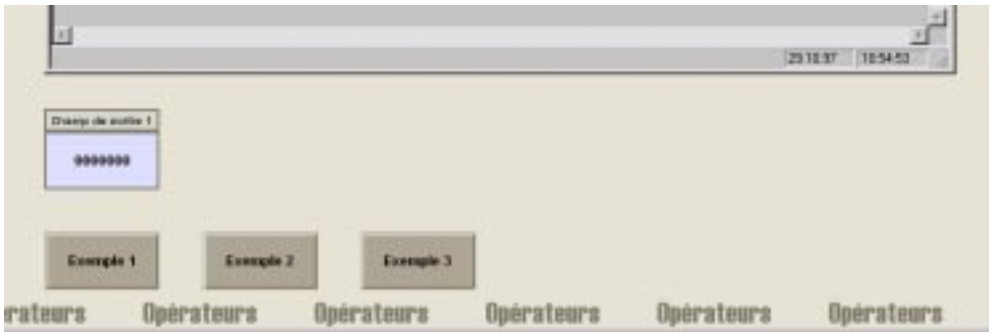
Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- Déclaration de variable.
- Dans la section *set tags* , la valeur de la variable est modifiée dans la *condition de test if*.
- _La nouvelle valeur est affectée à la variable WinCC interne (*S32i_course_tag_02*) par la fonction WinCC prévue à cet effet *SetTagDWord*.
- Pour l'objet *rectangle_01* cette variable est liée à la *positionX*. Le rafraîchissement de la position s'effectue lors des modifications de cette variable *S32i_course_tag_02*.
- La valeur est alors sortie avec la fonction *printf* dans la fenêtre de diagnostic.

4.3 Opérateurs et fonctions mathématiques en C

Vous pouvez, dans notre projet, accéder aux exemples sur les opérateurs en sélectionnant le bouton *Opérateurs* . Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.

Vue des opérateurs



Un clic de la souris sur le bouton *Opérateurs* permet d'afficher la vue ci-dessus. Les différents boutons de la figure correspondent aux exemples décrits. Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.

Un peu de théorie sur les opérateurs avant d'examiner nos exemples.
Les *opérateurs* gèrent ce qui se fait avec les *variables* et *constantes*.
On utilise les *opérateurs* pour combiner entre elles des *variables* et des *constantes* afin d'obtenir de nouveaux contenus de *variable*.

Les opérateurs sont divisés en différents types. Les plus importants pour nous sont indiqués ci-dessous :

Opérateurs arithmétiques

Type	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division
++	Incrémentation
--	Décrémentation

Les opérateurs arithmétiques sont par exemple utilisés dans le *dialogue dynamique* pour la mise en relation de plusieurs variables (p. ex. état moteur 1 et état moteur 2). Les opérateurs arithmétiques s'emploient également pour écrire des formules dans des *actions C*.

Opérateurs logiques et de comparaison

Type	Description
>	plus grand
>=	plus grand ou égal
==	Egal
!=	non égal
<=	plus petit ou égal
<	plus petit
&&	ET logique
||	OU logique
!	Négation logique

Les opérateurs arithmétiques et logiques sont nécessaires pour la mise en relation de plusieurs variables dans le *dialogue dynamique* (p. ex. état moteur et !panne).
Les opérateurs logiques figurent dans les interrogations des *actions C*.

Opérateurs de bits

Type	Description
&	Connexion binaire ET
|	Connexion binaire OU
^	OU exclusif
<<	Décalage de bits vers la gauche
>>	Décalage de bits vers la droite

Les opérateurs binaires sont par exemple utilisés dans les *actions C* pour interroger l'état de bit ou pour positionner les bits dans un mot de données (p. ex. mettre à 1 le bit d'état moteur dans le mot à la position 1, état moteur | 0x0002).

4.3.1 Algèbre booléenne

Fonctions logiques

En mathématique, les systèmes de variables logiques liés par des fonctions logiques font partie de ce que l'on appelle **l'algèbre booléenne**.
Lors de la création d'*actions en C*, il est fréquent que des variables logiques soient liées par des fonctions logiques pour former de nouvelles variables.

Toute fonction logique peut être représentée dans une **table de fonctions**. Cette table de fonctions s'appelle également **table de vérité**.
Pour représenter des combinaisons logiques, on utilise des symboles spécifiques standardisés. Ces symboles s'appellent également des **portes**.

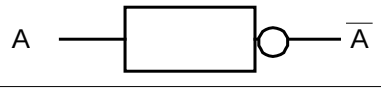
4.3.1.1 Fonctions logiques de base

Sont représentée ci-après, les principales fonctions logiques de base. Les fonctions logiques sont souvent désignées par leur appellation anglaise.

Inverterer

L'**inverseur** assure la fonction de négation du signal d'entrée. La négation est symbolisée par le cercle à la sortie de la boîte.

a	Not
0	1
1	0



Fonction ET

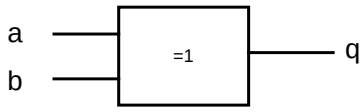
La **fonction ET** n'est à un que si les deux variables sont à un.

a	b	And
0	0	0
0	1	0
1	0	0
1	1	1

Fonction OU

La **fonction OU** n'est à un que si au moins l'une des deux variable est à un.

a	b	Or
0	0	0
0	1	1
1	0	1
1	1	1



Fonction NON-ET

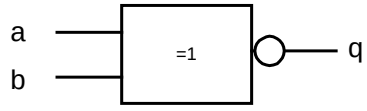
Une fonction ET suivie d'une négation s'appelle une **fonction NON-ET**. Cette fonction prend la valeur zéro, lorsque les deux variables sont à un.

a	b	Nand
0	0	1
0	1	1
1	0	1
1	1	0

Fonction NON-OU

Une fonction OU suivie d'une négation s'appelle une **fonction NON-OU**. Cette fonction ne prend la valeur un que si les deux variables sont à zéro.

a	b	Nor
0	0	1
0	1	0
1	0	0
1	1	0



Fonction OU exclusif

Cette fonction n'est à un que si l'une ou l'autre des deux variables est à un. Cette fonction s'appelle une **fonction OU exclusif**.

a	b	EXOR
0	0	0
0	1	1
1	0	1
1	1	0

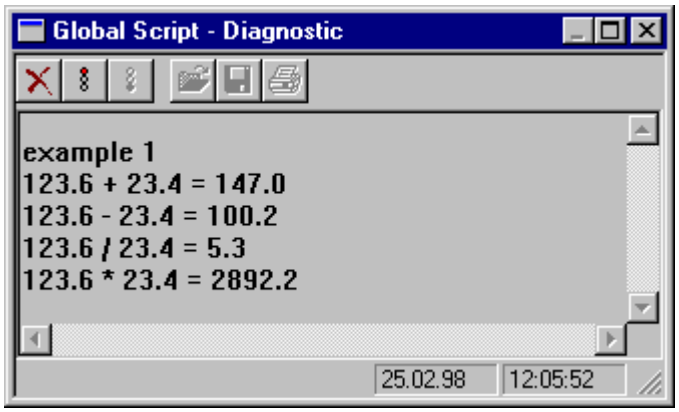
4.3.2 Exemple 1 - Utilisation des opérateurs des opérations arithmétiques de base

Nous allons montrer dans notre exemple1 comment utiliser les différents types d'opérateur correspondants aux opérations arithmétiques de base dans une *action C*.
L'action est configurée sur le bouton *exemple_01* →*Evénement* →*Souris* →*Cliquez bouton gauche*.
Lorsque le bouton *Exemple 1* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
float a,b;
float result1,result2,result3,result4;

a=123.6;
b=23.4;

result1=a+b;     // add
result2=a-b;     // subtract
result3=a/b;     // divide
result4=a*b;     // multiply

//output in diagnostics window
printf("\r\nexample 1\r\n");
printf("%.1f + %.1f = %.1f\r\n",a,b,result1);
printf("%.1f - %.1f = %.1f\r\n",a,b,result2);
printf("%.1f / %.1f = %.1f\r\n",a,b,result3);
printf("%.1f * %.1f = %.1f\r\n",a,b,result4);
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C.
- Affectation des valeurs aux variables.
- Exécution des opérations mathématiques.
- Les valeurs sont alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.

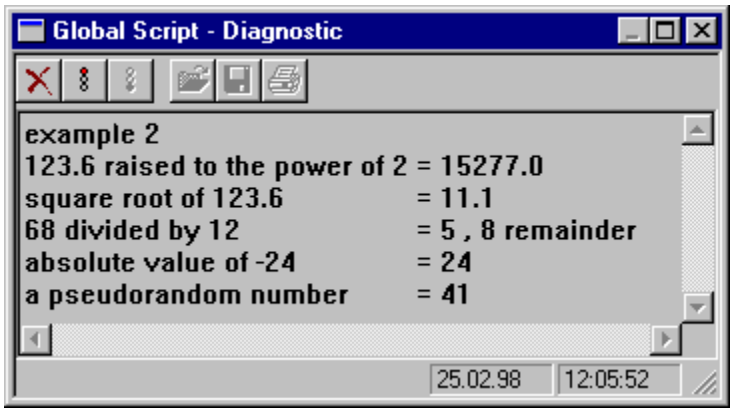
4.3.3 Exemple 2 - Fonctions mathématiques

Nous montrerons dans cet exemple 2 comment on utilise les fonctions mathématiques dans une *action C*.
L'action est configurée sur le bouton *exemple_02* → *Evénement* → *Souris* → *Cliquez bouton gauche* .
Lorsque le bouton *Exemple 2* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
float  a;
int    b,c,d;
div_t  result5;
double result1,result2;
int    result3,result4;

a=123.6;
b=-24;
c=68;
d=12;

result1=pow(a,2);
result2=sqrt(a);
result3=abs(b);
result4=rand();
result5=div(c,d);

//output in diagnostics window
printf("\r\nexample 2\r\n");
printf("%.1f raised to the power of\t = %.2f\r\n",a,result1);
printf("square root of %.1f\t = %.1f\r\n",a,result2);
printf("%d divided by %d\t\t = %d , %d
remainder\r\n",c,d,result5.quot,result5.rem);
printf("absolute value of %d\t = %d\r\n",b,result3);
printf("a pseudorandom number\t = %d\r\n",result4);
}
```

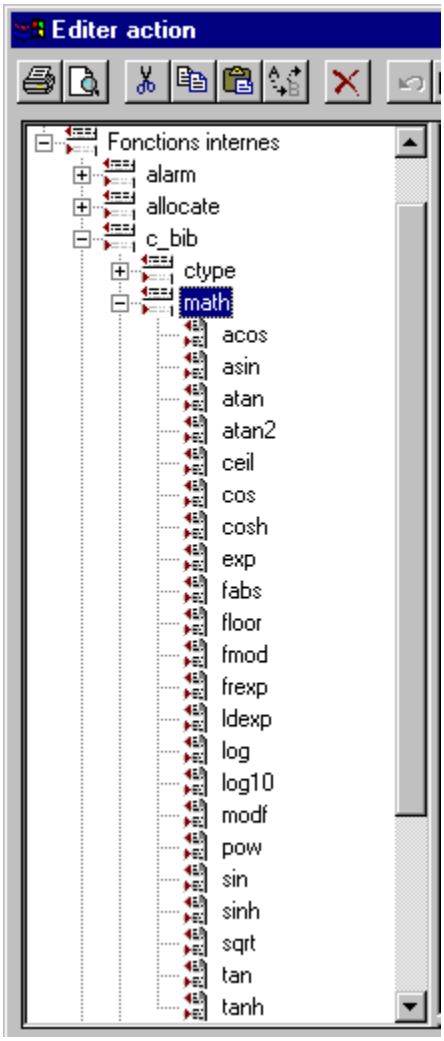
Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C.
- Des valeurs correspondantes sont affectées aux variables.
- Exécution des fonctions mathématiques. Vous trouverez ces fonctions avec *→fonctions internes→c_bib →math*
- Les valeurs sont alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.

Les fonctions mathématiques sont utilisables lorsqu'elles peuvent être sélectionnées. Par exemple, pour la formulation de l'interrogation dans le *dialogue dynamique* ou dans les *actions C*.



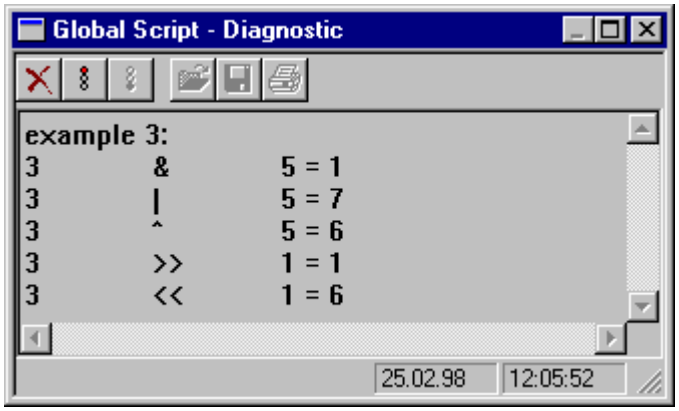
4.3.4 Exemple 3 - Opérateur pour opérations binaires

Nous vous montrerons dans notre exemple 3 comment on utilise les opérateurs de bits dans une *action C*.
L'action est configurée sur le bouton *exemple_03* →*Evénement* →*Souris* →*Cliquez bouton gauche* .
Lorsque le bouton *Exemple 3* est cliqué avec la souris , le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
int a,b,c;
int result1,result2,result3,result4,result5;

a=3;  // binary 011
b=5;  // binary 101

result1=a&b; // AND
result2=a|b; // inclusive OR
result3=a^b; // exclusive OR
result4=a>>1;// shift right
result5=a<<1;// shift left

//write result of OR in output_01
SetTagDWord("S32i_course_op_03",result2);

//output in diagnostic window
printf("\r\nexample 3:\r\n");
printf("%d\t&\t%d = %d\r\n",a,b,result1);
printf("%d\t|\t%d = %d\r\n",a,b,result2);
printf("%d\t^\t%d = %d\r\n",a,b,result3);
printf("%d\t>>\t1 = %d\r\n",a,result4);
printf("%d\t<<\t1 = %d\r\n",a,result5);
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C et l'affectation des valeurs.
- Exécution des opérations sur bits
- Sortie du résultat de l'opération logique *OU inclusif* dans le *champ d'affichage 1*.
- Les valeurs sont alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.

4.4 Pointeur en C

Dans notre projet, vous pouvez accéder aux exemples sur les pointeurs en sélectionnant le bouton *Pointeur dans la vue synoptique.* Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.

Vue de pointeur

Un clic de la souris sur le bouton *Pointeur* permet d'afficher la vue ci-dessus. Les différents boutons de la figure correspondent aux exemples décrits. Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.

Avant d'examiner nos exemples, un peu de théorie sur les pointeurs .
Les pointeurs sont des éléments essentiels des langages de programmation C et C++.

Définition et notation des pointeurs

Un pointeur ne contient pas la valeur d'une variable mais **pointe** simplement **sur l'adresse** à laquelle la valeur de la variable est rangée.
Le pointeur doit posséder le même type de données que la variable sur laquelle il pointe !
Un **astérisque placé devant** une variable déclare toujours cette variable comme pointeur.
L'opérateur * **permet d'accéder à la valeur de la variable sur laquelle pointe le pointeur. On le désigne également comme opérateur de contenu.**
Celui-ci ne doit cependant pas être confondu avec l'opérateur de multiplication. Il s'agit de deux opérateurs différents !
Le compilateur fait la distinction entre un opérateur de pointage et une multiplication à l'aide du contexte.

Extrait de programme de déclaration des pointeurs

```
{
intnumber;
int *pointer1;

 number1=123;
pointer1 = &number1;
}
```

L'exemple montre le mieux ce qui est un pointeur :

- Nous créons la variable *numéro1* et le pointeur *pointeur1*.
- La valeur *123* est affectée à la variable *numéro1*.
- L'**adresse** de la variable *numéro1* est affectée au pointeur *pointeur1*.
- Ceci est réalisé à l'aide de l'*opérateur d'adressage &*!
- Le *pointeur1* contient maintenant l'adresse de la variable *numéro1*.

L'avantage est que les pointeurs contiennent maintenant une adresse et pas de variable. Leur manipulation est donc plus souple et rapide.

Chaîne de caractères (String) comme pointeur

Lorsqu'une chaîne de caractères (string) est créée par un pointeur, la chaîne de caractères est écrite quelque part dans la mémoire. Le pointeur pointe le premier élément de la chaîne de caractères. Ceci fonctionne certes mais **relève d'un style de programmation mauvais et dangereux** !

Une chaîne de caractères –String – est caractérisée en C par le début de la chaîne de caractères (adresse de début de la chaîne de caractères) et par l'élément final (caractère nul, \0) :

W	i	n	C	C	\0

Dans la configuration de WinCC, vous rencontrerez des pointeurs par exemple en liaison avec les paramètres de fonctions :
lpszPictureName est par exemple un pointeur sur le début du nom de la figure (p. ex. Start),

S	t	a	r	t	\0

lpszObjectName est un pointeur sur le début du nom de l'objet (p. ex. Cercle 1) .

K	r	e	i	s	1	\0

La véritable utilité des pointeurs apparaîtra nettement lorsque nous utiliserons des variables du type **tableaux .** Ces tableaux sont appelés **Arrays** en anglais.
On peut désigner par conséquent des groupes ou des suites de variables d'un même type par le nom Tableau.

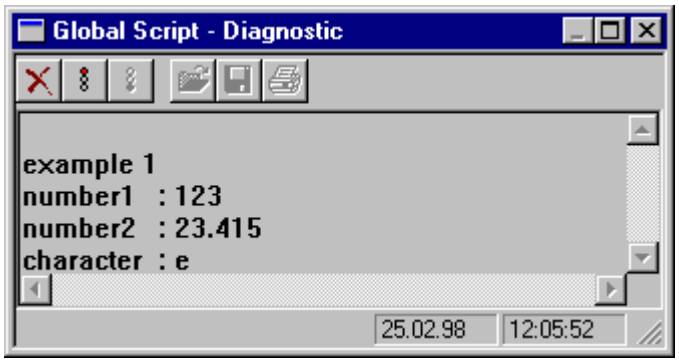
4.4.1 Exemple 1 - Pointeurs

Nous montrerons dans notre exemple 1 comment utiliser les pointeurs en C.
L'action est configurée sur le bouton *exemple_01* →*Evénement* →*Souris* →*Cliquez bouton gauche*.
Lorsque le bouton *Exemple 1* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//declaration of tags
intnumber1;
float number2;
char  character;

//declaration of pointer
int *pointer1;
float *pointer2;
char  *pointerc;

//set tag values
 number1  =123;
 number2  =23.415;
character = 'e';

//set adresses
pointer1= &number1;
pointer2=&number2;
pointerc=&character;

//set internal tag
SetTagDWord("S32i_course_point_00",*pointer1);

//output in diagnostics window
printf("\r\nexample 1\r\n");
printf("number1 = %d\r\n",GetTagDWord("S32i_course_point_00"));
printf("number2 = %.3f\r\n",*pointer2);
printf("character  = %c\r\n",*pointerc);
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C.
- La déclaration des pointeurs.
- Affectation des valeurs aux variables.
- Dans cette section, les *adresses* des variables sont affectées aux *pointeurs*.
- Les valeurs sont alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.

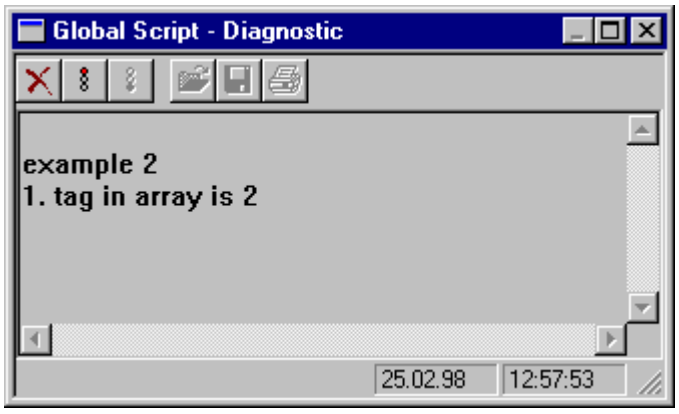
4.4.2 Exemple 2 - Pointeurs en liaison avec des variables de WinCCpointeur

Nous montrerons dans notre exemple 2 l'utilisation de pointeurs en liaison avec des *variables internes*. L'action est configurée sur le bouton *exemple_02* →*Evénement* →*Souris* →*Cliquez bouton gauche* . Lorsque le bouton *Exemple 2* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//declaration of array
int array[4];

int index;
int *pointer;

//set tag values
array[0]=2;
array[1]=4;
array[2]=6;
array[3]=8;

//pointer to first tag of array
pointer=&array[0];

//read index from input_01
index=GetTagDWord("S32i_course_point_04");
index--;

//set internal tag
SetTagDWord("S32i_course_point_03",*(pointer+index));

//output in diagnostics window
printf("\r\nexample 2\r\n");
printf("%d. tag in array is %d\r\n",(index+1),*(pointer+index));
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration d'un tableau (quatre entiers).
- La déclaration d'une variable.
- La déclaration d'un pointeur.
- Affectation des valeurs au tableau.
- Dans cette section, l'adresse du premier champ du tableau est affectée au pointeur.
- Lire l'index voulu dans le champ de saisie 1 et le mettre au format C correct (le premier champ du tableau possède l'index 0).
- Additionner l'index courant au pointeur de la première position du tableau. Le pointeur ainsi déterminé permet de lire le champ voulu.
- Les valeurs sont alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.

4.4.3 Exemple 3 - Pointeurs en liaison avec l'exécution de chaînes de caractères

Nous montrerons dans notre exemple 3 comment utiliser les pointeurs en C. Dans cet exemple, nous modifierons la représentation d'un texte sous forme d'un bandeau.
L'action est configurée sur le texte statique *static text_01* → *Propriétés* → *Police* → *Texte* . Lorsque vous cliquez sur le bouton *Exemple 3* avec la souris, la variable *BINi_varia_point_05* est mise à *1* par la liaison directe. Le script ci-dessous est exécuté toutes les 250 ms. Le test if n'est exécuté que si le bouton *Exemple 3* est enfoncé.

Action C

```
#include "apdefap.h"
 long _main(char* lpszPictureName, char* lpszObjectName, char*
                 lpszPropertyName)
{
//declaration of static tag
staticint i = 10;

char *str = "              WinCC";

if (GetTagBit("BINi_varia_point_05")) {
       if (i>19) (i=0);//check limit
       i++;//inc
       }
//return string
return(str+i);
}
```

Les conséquences sont directement visibles dans la vue :

Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables comme variables *statiques.* Les variables précédées de 'static' conservent leur valeur durant la sélection de la vue. L'initialisation n'a lieu que lors du premier appel de l'action.
- La déclaration du pointeur et l'affectation de valeurs.
- Vérification que le bouton *Exemple 3* est bien enfoncé.
- Vérification que la position ne dépasse pas la longueur maximale de la chaîne de caractères. Si c'est le cas, on recommence l'opération.
- Incrémentation de la position.
- Incrémentation de la position du pointeur sur la chaîne de caractères et renvoi du résultat par *return* .

4.5 Boucles et instructions conditionnelles en C

Dans notre projet, vous pouvez accéder aux exemples sur les boucles en sélectionnant le bouton *Boucles*. Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.

Vue de boucles

Un clic de la souris sur le bouton *Boucles* permet d'afficher la vue ci-dessus. Les différents boutons de la figure correspondent aux exemples décrits. Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.
Ü et *3* sont d'autres exemples de boucles.

Boucles

Avant d'examiner nos exemples, un peu de théorie sur les boucles.
Il existe d'une manière générale deux types de boucles : Les boucles avec *test en début de boucle* et les boucles avec *test en fin de boucle*.
Les boucles avec *test en début de boucle* vérifient **avant** l'exécution du corps de la boucle et les boucles avec *test en fin de boucle* **après** l'exécution du corps de boucle si la condition de parcours de la boucle est remplie. C'est-à-dire que les boucles avec test de condition en fin de boucle sont toujours parcourues au moins 1x.

Lorsque le corps de la boucle est constitué d'une instruction, les accolades peuvent être omises.

while

```
while ([Condition])
      { Corps de boucle }
```

Le corps de la boucle est répété tant que la *condition* est remplie.

do - while

```
do
      { Corps de boucle }
while [Condition] ;
```

Le corps de boucle est exécuté au moins une fois et répété jusqu'à ce que la condition ne soit plus remplie.

Nota :
Pour les boucles avec *test en fin de boucle*, l'instruction contenant la condition while est **toujours** suivie d'un point-virgule ; le point-virgule peut être **omis** par contre pour les boucles avec *'test en début de boucle'* .

for

```
for ( [ Init ] ; [ Condition ] ; [ Instruction ] )
      { Corps de boucle }
```

Tout ceci a l'air très crypté. Nous allons donc transformer tout cela pour obtenir une expression équivalente :

```
Init;
while [Condition]
      {Corps de boucle;
      [Instruction];
      }
```

Nota :
Notez que les boucles peuvent devenir des boucles sans fin si la condition est **toujours remplie**. Vous devez par conséquent tester vos *Actions C* immédiatement après les avoir formulées afin de vérifier si la séquence est exempte d'erreur.

Instructions conditionnelles

Dans les boucles, un corps de boucle est répété aussi longtemps que la condition est remplie.
Dans les instructions conditionnelles, l'instruction est exécutée exactement une fois si la condition est remplie.
Faire attention aux types de variables dans les comparaisons !

if

```
if  ( [Condition1] )
      { Corps1 }
else
      { Corps2 }
```

Le corps d'instruction1 est exécuté si la condition est remplie (valeur différente de 0).
Si la condition n'est pas remplie (la valeur est 0), c'est le corps d'instruction2 qui est exécuté en alternative.
else peut être également omis ; aucune alternative n'est alors exécutée.

La condition *if* n'entraînant que le contrôle de la valeur numérique d'une expression, il est possible d'utiliser des abréviations. Les deux instructions ci-dessous sont par conséquent identiques.

```
if  (Condition)
if  (Condition != 0)
```

Il est possible de combiner plusieurs tests de condition :

```
if  ( [Condition1] )
      { Corps1 }
else if ( [Condition2] ) // deuxième interrogation if (imbriquée)
      { Corps2 }
else
      { Corps3 }
```

Dans cet exemple, la première condition est testée. Si la condition n'est pas réalisée, la seconde condition est testée. Si la condition est réalisée, le corps d'instruction2 est exécuté. Si aucune des conditions n'est remplie, le corps d'instruction3 est exécuté.

Nota :
Si une condition est remplie, les autres cas ne sont plus traités.

Lorsqu'il y a plus de deux interrogations, il est préférable d'utiliser les instructions switch et case.

switch & case

```
switch ( [Variable] ) {
      case [Terme1] :
            Action1;
           break;
      case [Terme2] :
            Action2;
           break;
     default :
            Action3;
           break;
}
```

On vérifie ici l'égalité d'une variable. L'instruction *switch* permet d'indiquer la variable à tester. Le programme vérifie tout d'abord si la variable est égale au *terme1* ; si c'est le cas, *action1* est exécutée. Cette action peut contenir un nombre quelconque d'instructions.
Les actions se terminent par un break!

Le mot-clé *default* n'est pas suivi d'une comparaison. Ces actions sont exécutées lorsqu'aucune des conditions *case* n'est remplie. Il s'agit donc d'une instruction à exécuter par défaut aussi longtemps que les conditions d'un autre cas ne sont pas remplies.
L'*instruction break* peut être omise avec l'instruction *default* (toujours la dernière instruction), mais ce n'est pas de la programmation élégante.

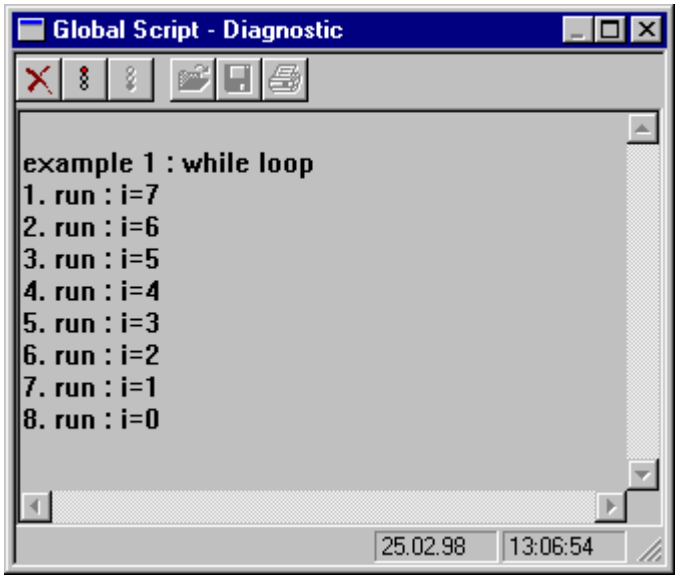
4.5.1 Exemple 1 - Boucle do - while

Nous montrerons dans l'exemple 1 l'utilisation de la boucle *while* dans une *action C*.
L'action est configurée sur le bouton *exemple_01* → *Evénement* → *Souris* → *Cliquez bouton gauche*.
Lorsque le bouton *Exemple 1* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//while-loop
int i,j;

printf("\r\nexample 1 : while loop\r\n");

j=0;  // execution count
i=8;  // tag used for loop action

while (i>0) {
    j++;
    i--;
    printf("%d. run : i=%d\r\n",j,i);
}//while
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C. La lettre *i* est utilisée pour la fonction de boucle et la lettre *j* pour afficher le nombre de passages de la boucle.
- Affichage dans la fenêtre de diagnostic de ce qui se passe ensuite.
- Les variables sont initialisées pour l'exécution de la boucle.
- Exécution de la boucle *while* tant que *i>0*.
- La variable *j* est incrémentée, la variable *i* décrémentée.
- La valeur courante des variables est alors sortie avec la fonction *printf* dans la fenêtre de diagnostic.

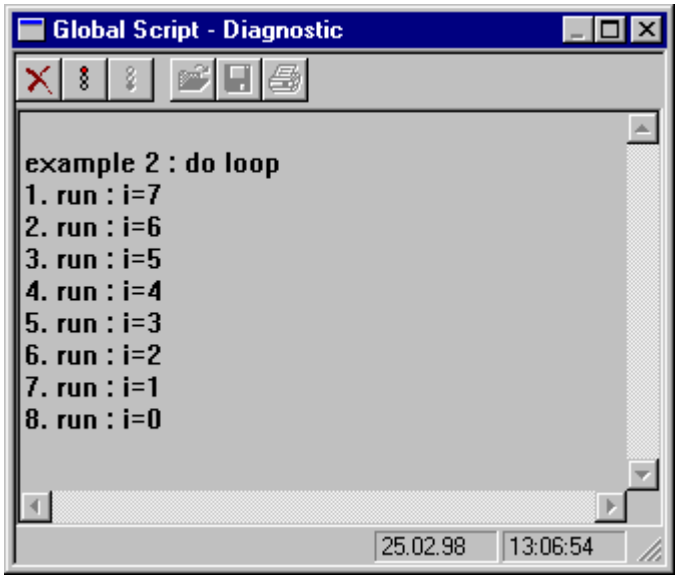
4.5.2 Exemple 2 - Boucle do - while

Nous montrerons dans l'exemple 2 l'utilisation de la boucle *do - while* dans une *action C*.
L'action est configurée sur le bouton *exemple_02* → *Evénement* → *Souris* → *Cliquez bouton gauche* .
Lorsque le bouton *Exemple 2* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//do loop
int i,j;

printf("\r\nexample 2 : do loop\r\n");

j=0;  // execution count
i=8;  //tag used in loop action

do  {
    j++;
    i--;
    printf("%d. run : i=%d\r\n",j,i);
}//do
while (i>0);
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C. La lettre *i* est utilisée pour la fonction de boucle et la lettre *j* pour afficher le nombre de passages de la boucle.
- Affichage dans la fenêtre de diagnostic de ce qui se passe ensuite.
- Les variables sont initialisées pour l'exécution de la boucle.
- Exécution de la boucle *do - while* tant que *i>0* (1 passage au moins).
- La variable *j* est incrémentée, la variable *i* décrémentée.
- La valeur courante des variables est alors sortie avec la fonction *printf* dans la fenêtre de diagnostic.

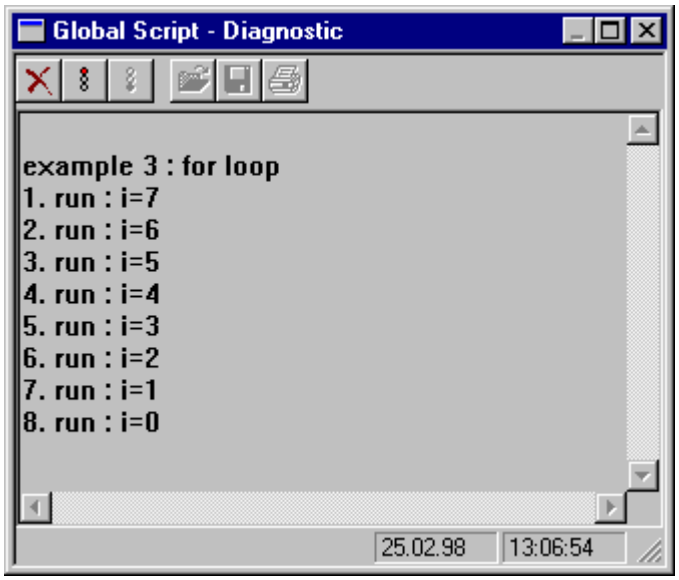
4.5.3 Exemple 3 - Boucle for

Nous montrerons dans l'exemple 3 l'utilisation de la boucle *for* dans une *action C*.
L'action est configurée sur le bouton *exemple_03* → *Evénement* → *Souris* → *Cliquez bouton gauche* .
Lorsque le bouton *Exemple 3* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//for loop
int i,j;

printf("\r\nexample 3 : for loop\r\n");

j=0;  // execution count

for(i=7;i>=0;i--) {
    j++;
    printf("%d. run : i=%d\r\n",j,i);
}//for
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C. La lettre *i* est utilisée pour la fonction de boucle et la lettre *j* pour afficher le nombre de passages de la boucle.
- Affichage dans la fenêtre de diagnostic de ce qui se passe ensuite.
- La variable de compteur de boucles est initialisée.
- Exécution de la boucle *for* tant que *i>=0*. La valeur initiale de *i* est *7*.
- La variable *j* est incrémentée à l'exécution de la boucle, la variable *i* est décrémentée à chaque passage.
- La valeur courante des variables est alors sortie avec la fonction *printf* dans la fenêtre de diagnostic.

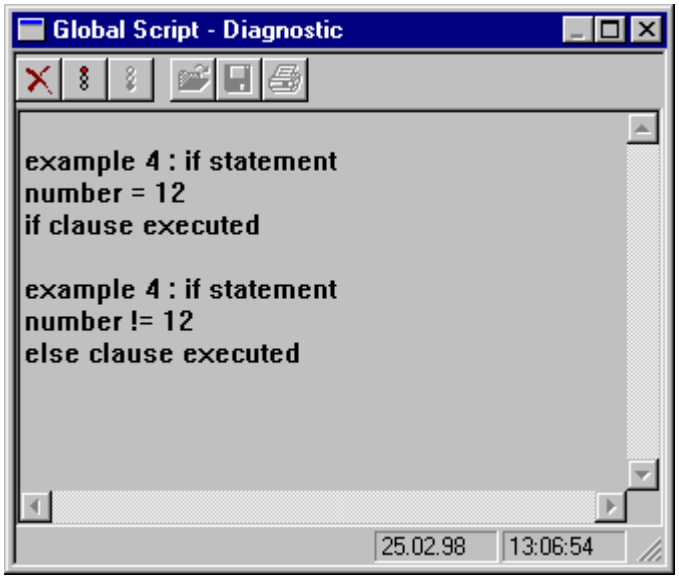
4.5.4 Exemple 4- Instructions conditionnelles avec if

Nous montrerons dans l'exemple 4 l'utilisation de l'instruction conditionnelle *if* dans une *action C*.
L'action est configurée sur le bouton *exemple_04* → *Evénement* → *Souris* → *Cliquez bouton gauche*.
Lorsque le bouton *Exemple 4* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//if statement
staticint number=12;

printf("\r\nexample 4 : if statement\r\n");

if (number==12) {
    printf("number = 12\r\n");
    printf("if clause executed\r\n");
    number=11;
}//if
else {
    printf("number != 12\r\n",number);
    printf("else clause executed\r\n");
    number=12;
}//else
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C comme variables *statiques*. La valeur de cette variable reste constante jusqu'au prochain changement de vue.
- Affichage dans la fenêtre de diagnostic de ce qui se passe ensuite.
- Exécution du *Test*.
- Si la condition *(numéro==12)* est remplie, la branche *if* est exécutée. Si la condition n'est pas remplie, c'est la branche *else* qui est exécutée.
- Le résultat courant du test est alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.

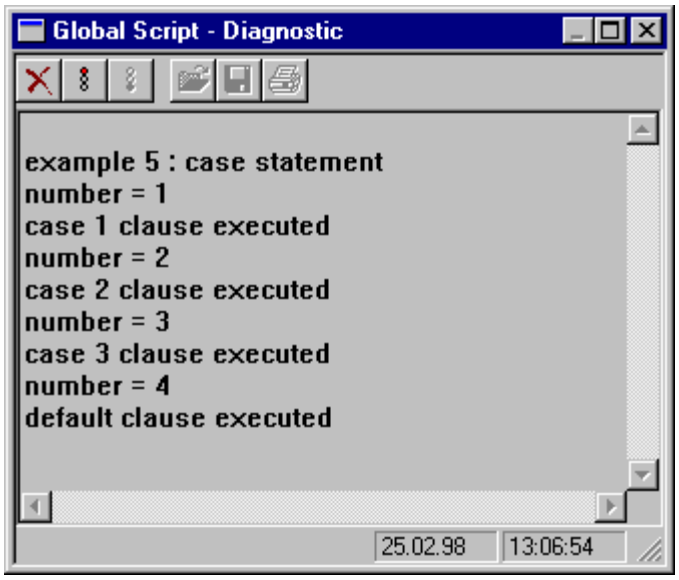
4.5.5 Exemple 5- Instruction conditionnelle avec switch et case

Nous montrerons dans l'exemple 5 l'utilisation de l'instruction conditionnelle *switch case* dans une *action C*.-
L'action est configurée sur le bouton *exemple_05* → *Evénement* → *Souris* → *Cliquez bouton gauche'* .
Lorsque le bouton *Exemple 5* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//case statement
staticint number=1;

if (number==1) printf("\r\nexample 5 : case statement\r\n");

switch (number) {
    case 1:{
    printf("number = %d\r\n",number);
    printf("case 1 clause executed\r\n");
    number++;
          break;
    }//case1
    case 2: {
    printf("number = %d\r\n",number);
    printf("case 2 clause executed\r\n");
    number++;
          break;
    }//case2
    case 3: {
    printf("number = %d\r\n",number);
    printf("case 3 clause executed\r\n");
    number++;
          break;
    }//case3
    default: {
    printf("number = %d\r\n",number);
    printf("default clause executed\r\n");
    number=1;
          break;
} //default
}//switch
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C comme variables *statiques*. La valeur de cette variable reste constante jusqu'au prochain changement de vue.
- Au premier passage, affichage dans la fenêtre de diagnostic de ce qui se passe ensuite.
- L'instruction *switch* définit la variable à tester *numéro*.
- Les instructions *case* vérifient si la condition est remplie. L'égalité de la variable est vérifiée. Si la condition n'est pas remplie, la condition *case* suivante est exécutée. Si la condition de cette branche n'est pas remplie, la condition *case* suivante est exécutée, etc.
- Une instruction *case* ou une instruction *'default'* est exécutée chaque fois que vous cliquez sur le bouton avec la .
- La fonction *printf* affiche le résultat dans la fenêtre de diagnostic.
- La valeur de la variable *numéro* est modifiée.

4.5.6 Exemple 6- Utilisation de variables statiques avec une instruction conditionnelle et valeur retournée

Nous montrerons dans l'exemple 6 l'utilisation de variables C du type *statique* en liaison avec une instruction conditionnelle. La modification de la taille de police est passée par la valeur retournée dans une *ActionC*.
L'action est configurée sur la propriété statique *static text_01 → Propriétés → Police → Taille de police*. Lorsque la vue est sélectionnée, le script ci-dessous est exécuté toutes les 250 ms.

Action C

```
#include "apdefap.h"
 long _main(char* lpszPictureName, char* lpszObjectName, char*
                 lpszPropertyName)
{
staticint i = 0;
static BOOL a=0;

if ((i<1)||(i>60)) a=!a;

//inc or dec i according to a
if (a) i=i+6;
else i-=6;//short form for i=i-6

//return
return(i+20);
}
```

Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C comme variables *statiques*. La valeur de cette variable reste constante jusqu'au prochain changement de vue. Les deux sont initialisées par *0*.
- La valeur de la variable *a* est déterminée. La valeur de la variable *a* est inversée chaque fois que *i* quitte la plage admissible.
- La variable *i* est incrémentée en fonction de la variable *a* ou décrémentée. Elle est incrémentée lorsque la condition est remplie. Autre notation possible : *if (a!=0)*.
- Pour le calcul de la taille de police, la taille minimale de 20 points est ajoutée à la variable *i* et le résultat passé par *Return* à la propriété *taille de police* de l'objet sous forme de valeur retournée .

4.5.7 Exemple 7- Utilisation de variables statiques avec une instruction conditionnelle et valeur retournée

Nous montrerons dans l'exemple 7 l'utilisation de variables C du type *statique* en liaison avec une instruction conditionnelle. La modification de la couleur de police est passée par la valeur retournée dans une *ActionC*.
L'action est configurée sur la propriété statique *static text_02 → Propriétés → Couleurs→ Couleur de police* . Lorsque la vue est sélectionnée, le script ci-dessous est exécuté toutes les 250 ms.

Action C

```
#include "apdefap.h"
 long _main(char* lpszPictureName, char* lpszObjectName, char*
                 lpszPropertyName)
{
staticlongint i=0;
static BOOL a=0;

if ((i<400)||(i>10000000)) a=!a;

//inc or dec i according to a
if (a==1) i+=500; //short form for i=i+500
else i=i-500;

//return color
return(i+100000);
}
```

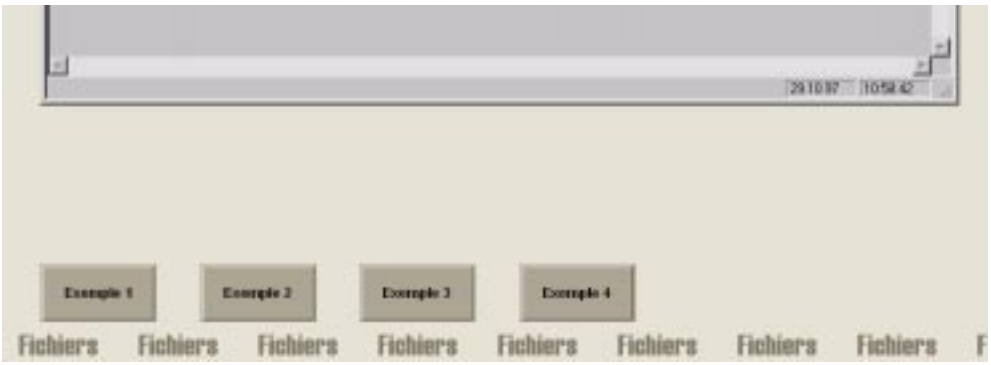
Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables C comme variables *statiques*. La valeur de cette variable reste constante jusqu'au prochain changement de vue.
- La valeur de la variable *a* est inversée chaque fois que *i* quitte la plage admissible.
- La variable *i* est incrémentée en fonction de la variable *a* ou décrémentée.
- Pour le passage de la valeur de la couleur, la valeur 100000 est ajoutée à la variable *i* et le résultat est passé à la propriété de l'objet. Vous trouverez des informations sur le codage des couleurs en annexe au chapitre *5.1.6 Table des couleurs*.

4.6 Opérations sur fichiers en C

Dans notre projet, vous pouvez accéder aux exemples sur les manipulations de fichier en sélectionnant le bouton *Fichiers.* Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.

Vue de fichiers



Un clic de la souris sur le bouton *Fichiers* permet d'afficher la vue ci-dessus. Les différents boutons de la figure correspondent aux exemples décrits. Le bouton droit de la souris **D** permet d'afficher le code source de l'exemple.

Avant d'examiner nos exemples, un peu de théorie sur les opérations sur fichiers.
Dans le langage C, un fichier est une collection de caractères indépendamment du contenu.
Il faut *ouvrir* un fichier avant de pouvoir le traiter en C (lecture, écriture).

fopen

La fonction *fopen* permet d'ouvrir un fichier. Le fichier à traiter et le mode de traitement sont passés comme argument.

```
FILE *fichier;
Fichier = fopen (nom de fichier, mode de travail);
```

Le nom de fichier peut également être transmis avec un chemin.
En cas d'erreur, aucune liaison à la description de fichier, c'est-à-dire en C à un *pointeur NUL,* est retournée. Les variables du type *FILE* contiennent les informations requises dans les instructions de fichier pour la manipulation de fichiers.

En mode d'édition, nous définissions les fonctions autorisées.

Mode	Description
R	Ouvrir le fichier pour lecture (r = read) La valeur retournée est NUL, lorsque le fichier n'existe pas ou lorsque l'opérateur ne possède pas le droit de lecture.
W	Ouvrir le fichier pour écriture (w = write) La valeur retournée est NUL, lorsqu'il s'agit d'un fichier READ ONLY (lecture seulement) ou lorsque le fichier n'existe pas.
A	Ouvrir le fichier pour l'ajouter à la fin (a = append) Lorsque le fichier existe, le texte est ajouté à la fin. Le fichier est créé s'il n'existe pas. La valeur retournée est NUL, lorsque la création d'un fichier n'est pas permise ou que ce fichier n'est pas accessible en écriture.
r+	Le fichier est ouvert pour lecture et écriture en alternance. Ouvrir le fichier pour lecture (r = read) La valeur retournée est NUL, lorsque le fichier n'existe pas ou s'il n'existe pas de droits de lecture et d'écriture pour ce fichier.
w+	Création d'un nouveau fichier pour lecture et écriture en alternance. Si le fichier existe déjà, il est **effacé !** La valeur retournée est NUL, lorsque l'opérateur ne possède pas le droit nécessaire à cette action.
a+	Ouverture d'un fichier pour lecture ou écriture (ajouter à la fin). Le fichier est créé s'il n'existe pas. La valeur retournée est NUL, lorsqu'un fichier ne peut être ni lu ni écrit ou lorsque les droits d'accès à ce fichier sont insuffisants.

fclose

fclose permet de fermer les fichiers préalablement ouverts avec *fopen* .

```
fclose (fichier);
```

fscanf

Cette instruction a la même fonction que *scanf,* à la différence près que cette fonction indique de quel fichier proviennent les informations.

```
fscanf ( fichier, "%c", caractères);
```

fprintf

Cette instruction a le même effet que *printf,* à la différence que cette fonction indique dans quel fichier les informations vont être imprimées.

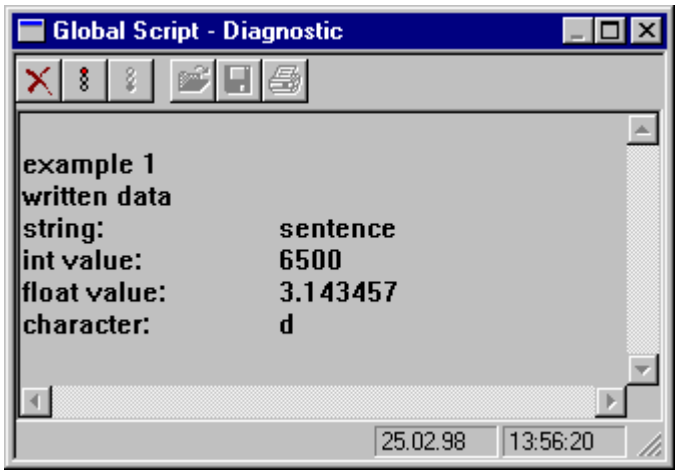
4.6.1 Exemple 1 - Ouvrir, écrire et fermer un fichier

Nous allons montrer dans l'exemple 1 les opérations fondamentales sur un fichier en C.
L'action est configurée sur le bouton *exemple_01* → *Evénement* → *Souris* → *Cliquez bouton gauche*.
Lorsque le bouton *Exemple 1* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
FILE   *fichier;
long   l;
float  fp;
char   s[20];
char   c;

//open file to write
fichier = fopen( "cours_00.dat", "w+" );

//file open error ??
if( fichier != NULL ){
       l=6500;
       strcpy(s,"sentence");
       fp=3.1434567;
       c='d';

       //write data
       fprintf( fichier,"%s\r\n%ld\r\n%f\r\n%c\r\n", s, l, fp, c );

       //output in diagnostics window
       printf("\r\nexample 1\r\n");
       printf("written data\r\n");
       printf("string:\t\t%s\r\n",s);
       printf("int value:\t\t%ld\r\n",l);
       printf("float value:\t%f\r\n",fp);
       printf("character:\t%c\r\n",c);

       //close file
       fclose (fichier);
       }//if
else printf( "file open error\r\n" );
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables et du pointeur.
- L'ouverture du fichier pour écrasement. Le fichier est créé s'il n'existe pas.
- Dans cette section, l'ouverture du fichier est vérifiée dans une interrogation conditionnelle. En cas de résultat positif, les valeurs sont affectées aux variables et écrites dans le fichier par *fprintf* . Si l'ouverture ou la création du fichier n'a pas fonctionné, la branche *else* est exécutée.
- Les valeurs sont alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.
- La dernière étape consiste à fermer le fichier.

4.6.2 Exemple 2 - Ouverture, extension et fermeture d'un fichier

Nous allons montrer dans l'exemple 2 les opérations fondamentales sur un fichier en C.
L'action est configurée sur le bouton *exemple_02* → *Evénement* → *Souris* → *Cliquez bouton gauche* .
Lorsque le bouton *Exemple 2* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                char* lpszPropertyName, UINT nFlags, int x, int y)
{
static int   i = 0;
FILE         *fichier;
char         s[200];

switch (i) {
      case 0 : strcpy(s,"This\r\n");   break;
      case 1 : strcpy(s,"sentence\r\n");     break;
      case 2 : strcpy(s,"is\r\n");     break;
      case 3 : strcpy(s,"a\r\n");            break;
      case 4 : strcpy(s,"long\r\n");   break;
      case 5 : strcpy(s,"one\r\n");    break;
      }

//open file to write and append data
fichier = fopen( "cours_00.dat", "a" );

//file open error ??
if( fichier != NULL ){

      //write data
      fprintf(fichier,"%s",s);

      //output in diagnostics window
      if (i==0;
            printf("\r\nexample 2");
            printf("\r\nwritten data\r\n");
            }
      printf("string:\t\t%s",s);

      fclose( fichier );//close file
      }//if
else printf("\r\nfile open error\r\n");
i++;
if (i>5) (i=0);
}
```

Affichage dans la fenêtre de diagnostic :

Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- La déclaration des variables et du pointeur.
- Génération d'un texte en fonction de la variable statique *i*.
- Ouverture du fichier pour le rajout de données en fin de fichier.
- Dans cette section, l'ouverture du fichier est vérifiée dans une interrogation conditionnelle. Si le fichier a été ouvert, le texte courant est inscrit dans le fichier par *fprintf*. Les données (contenu de la variable *s*) sont rajoutées en fin de fichier.
- Le texte écrit est alors sortie avec la fonction *printf* dans la fenêtre de diagnostic.
- La dernière étape consiste à fermer le fichier.

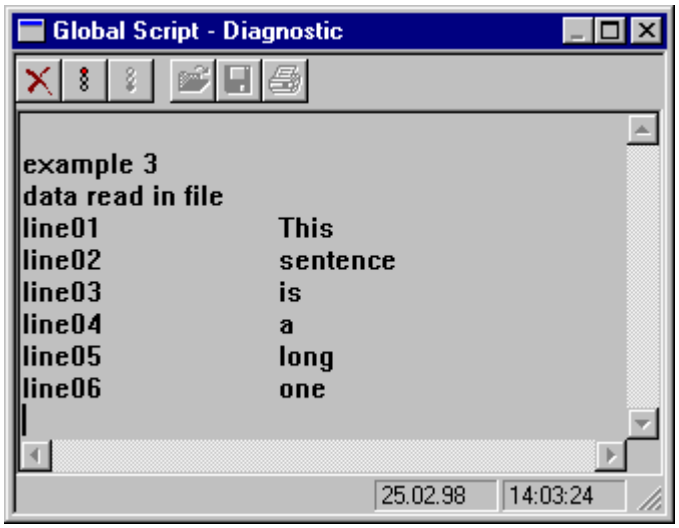
4.6.3 Exemple 3 - Ouvrir, lire et fermer un fichier

Nous allons montrer dans l'exemple 3 les opérations fondamentales sur un fichier en C.
L'action est configurée sur le bouton *exemple_03* → *Evénement* → *Souris* → *Cliquez bouton gauche* .
Lorsque le bouton *Exemple 3* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
FILE   *fichier;
int    i;
char   t[20];
char*  z;
//open file to read
fichier = fopen("cours_00.dat","r");

//file open error ??
if( fichier != NULL ){

printf("\r\nexample 3");
printf("\r\ndata read in file\r\n");

//read data in file
i=1;
while (i) {
       i++;
       z=fgets(t,20,fichier);
       if (z==NULL) break;//end of file
       //output in diagnostics window
       printf("line%02d\t\t%s",(i-1),t);
       }//while

//close file
       fclose (fichier);
       }//if
else {
   printf("\r\nfile open error\r\n" );
}//else
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- Déclaration des variables et pointeurs.
- Ouverture du fichier.
- Dans cette section, l'ouverture du fichier est vérifiée dans une interrogation conditionnelle. Si cela a fonctionné, les données sont lues ligne par ligne dans le fichier par *fgets* et affichées dans la fenêtre de diagnostic par la fonction *printf*. Dès que la fin du fichier est atteinte (valeur retournée égale à 0) la boucle *while* est quitté par *break* .
- Le fichier est enfin fermé.

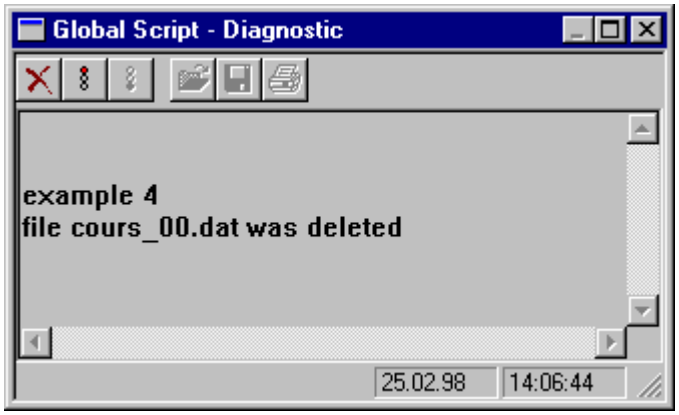
4.6.4 Exemple 4 - Supprimer un fichier

Dans l'exemple 4, nous montrons comment supprimer le fichier préalablement créé.
L'action est configurée sur le bouton *exemple_05* → *Evénement* → *Souris* → *Cliquez bouton gauche'* .
Lorsque le bouton *Exemple 5* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//delete file
remove("cours_00.dat");
printf("\r\n\r\nexample 4\r\n");
printf("file cours_00.dat was deleted\r\n");
}
```

Affichage dans la fenêtre de diagnostic :



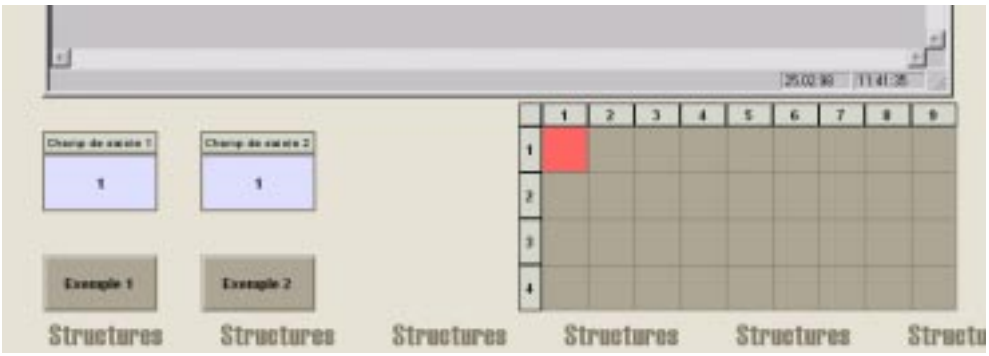
Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- Le fichier est supprimé.
- Affichage dans la fenêtre de diagnostic de ce qui se passe ensuite.

4.7 Structures en C

Dans notre projet, vous pouvez accéder aux exemples sur le sujet en sélectionnant le bouton *Structures.* Le bouton droit de la souris D permet d'afficher le code source de l'exemple.

Vue de structures



Un clic de la souris sur le bouton *Structures* permet d'afficher la vue ci-dessus. Les différents boutons de la figure correspondent aux exemples décrits. Le bouton droit de la souris D permet d'afficher le code source de l'exemple.
Les valeurs des *champs de saisie* de l'*exemple 2* sont exploitées dans la partie de droite.

Avant d'examiner nos exemples, un peu de théorie sur les structures.

Définition d'une structure

En C, une structure se déclare toujours de la manière suivante.

```
struct nom de structure    {
                    [ Eléments ]
                    };
```

Les éléments de la structure sont des variables définies constituant elles-mêmes les champs de la structure à laquelle elles appartiennent.
Vous pouvez manipuler une structure comme des types de variables car il s'agit d'une accumulation de types de données.
Avec

```
struct ganz   {
              int    entier;
              char caractère[50];
              };

struct ganz   structure[2];
```

nous définissons, dans la partie de la déclaration, la composition de la structure.
On peut imaginer que l'on crée un type de données *struct ganz* et que la variable *structure* est du type *struct ganz* .

La déclaration peut être simplifiée.

La déclaration de la composition de la structure et la définition d'une variable peuvent être réunies en une seule instruction.

```
struct ganz   {
              int entier;
              char caractère[50];
              } structure[2];
```

Accès aux variables dans la structure

Nom de structure.Nom de variable permet d'accéder sélectivement aux différentes données.
Il est important que les différents éléments soient séparés par un point. Ce point sépare les différents éléments dans la hiérarchie.
Le nom de variable est toujours le nom qui est utilisé dans la création de la structure !
Avec

```
structure[0].entier = 2;
strcpy(structure[0].caractères ,"notre première structure");
```

Les instructions ci-dessus affectent des valeurs aux éléments. Toujours veiller ce faisant que le type de données soit correct (*strcpy* permet d'affecter une chaîne de caractères à une séquence de caratères).
On peut accéder au champ dans les structures :

```
printf ("%c",structure[0].caractères[5] );
```

L'instruction ci-dessus nous affiche par exemple la 6e lettre de la chaîne de caractères du 1e champ de *structure [0]*. **Les indices commencent par 0 !**

Pointeurs sur des éléments de structure

Une structure peut contenir des types de variables quelconque. Ceci inclut les pointeurs.
La structure ci-dessous est par conséquent tout à fait imaginable :

```
struct structure {
              int   entier;
              char  *chaîne de caractères;
              }
```

Nota :
Lorsqu'un pointeur est utilisé sur une chaîne de caractères (p. ex. *chaîne de caractères), la mémoire doit être disponible ou avoir été préalablement allouée (p. ex. avec la fonction interne sysMalloc (8)).

Pointeurs sur des éléments de structure

Reprenons l'exemple de la structure de l'exemple ci-dessus et générons le pointeur suivant :

```
struct structure    *pointeur;
```

Si nous voulons accéder maintenant à un élément, nous devons écrire l'instruction

```
(*pointeur).entier;  // ou plus simplement  pointeur->entier;
```

écrire.

Nota :
Le point étant prioritaire sur l'opérateur *, la première expression doit être entre parenthèses.

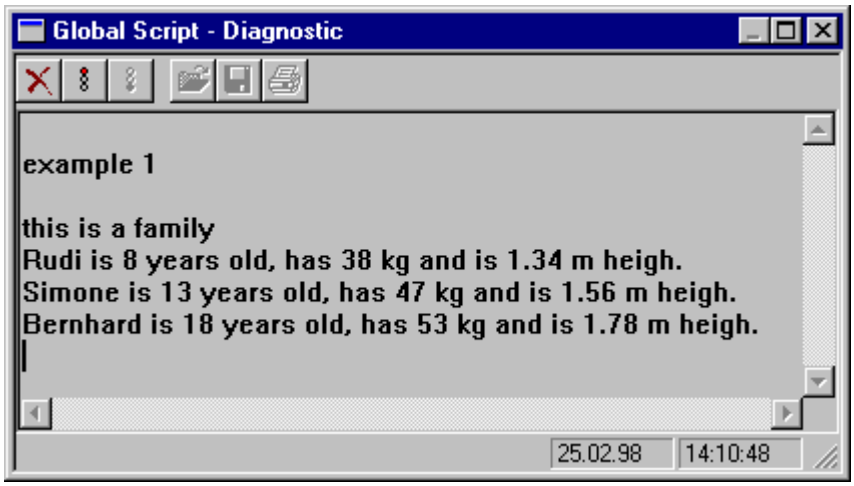
4.7.1 Exemple 1 - Structures en C

Nous montrerons dans l'exemple 1 comment utiliser les structures en C.
L'action est configurée sur le bouton *exemple_01* →*Evénement* →*Souris* →*Cliquez bouton gauche*.
Lorsque le bouton *Exemple 1* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//declare structure
struct daten {
       char name[40];
       int age;
       int weight;
       float height; }  family[3];

int i;

//set values to structure elements
strcpy(family[0].name,"Roger");
strcpy(family[1].name,"Simone");
strcpy(family[2].name,"Bernard");

family[0].age=8;
family[1].age=13;
family[2].age=18;

family[0].weight=38;
family[1].weight=47;
family[2].weight=53;

family[0].height=1.34;
family[1].height=1.56;
family[2].height=1.78;

//output in diagnostics window
printf("\r\nexample 1\r\n");
printf("\r\nthis is a family\r\n");
for (i=0;i<3;i++) {
   printf("%s is %d years old, has %d kg and is %.2f m
heigh.\r\n",family[i].name,family[i].age,family[i].weight,family[i].
height);
   }
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- Déclaration de la structure et de la variable.
- Affecter les valeurs aux champs de structure.
- Les valeurs (informations) sont alors sorties avec la fonction *printf* dans la fenêtre de diagnostic.

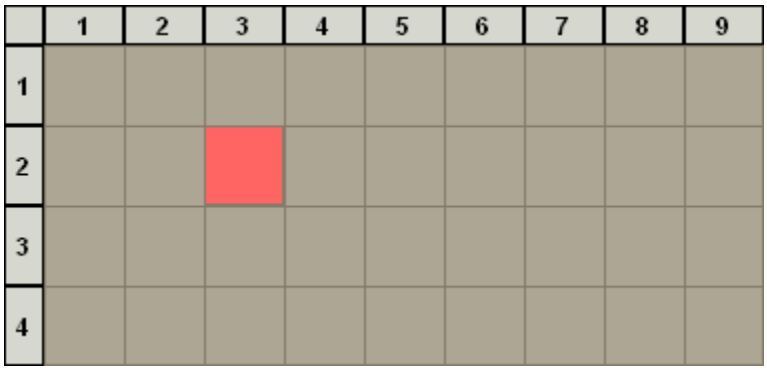
4.7.2 Exemple 2 - Structures en C en liaison avec WinCC

Nous montrerons dans l'exemple 2 comment utiliser les variables de type structure dans WinCC. Avant d'utiliser ces variables dans notre action, nous devons les créer dans la gestion des variables de *Control Center*. L'action est configurée sur le bouton *exemple_02* → *Evénement* → *Souris* → *Cliquez bouton gauche* . Lorsque le bouton *Exemple 2* est cliqué avec la souris, le script ci-dessous est édité.

Action C

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
int input1,input2;
int output1,output2;

//read selected position in input_01 and input_02
input1=GetTagDWord("S32i_course_str_01");
input2=GetTagDWord("S32i_course_str_02");

input1=input1-1;
input2=input2-1;

//calculate new rectangle position
output1=500+(input1*40);
output2=470+(input2*40);

//set internal tags which contain rectangle position
SetTagDWord("STUi_course_str_00.xpos",output2);
SetTagDWord("STUi_course_str_00.ypos",output1);
}
```

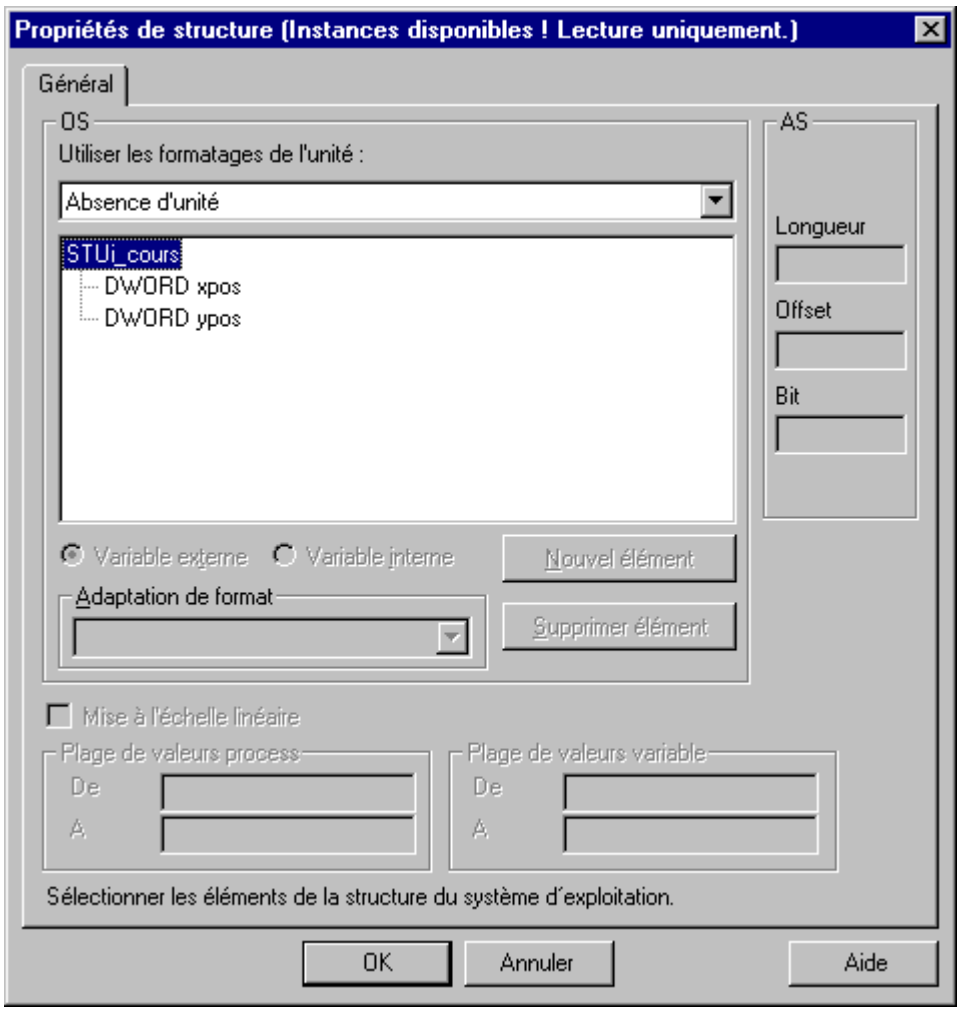
Affichage à l'écran :



Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- Déclaration de variable.
- Lecture des valeurs dans les champs d'E/S. Les limites d'introduction sont définies directement dans les *Champs E/S*.
- Convertir les indications de position entrées en positions à l'écran.
- Affectation des valeurs calculées aux variables de structure WinCC. Les valeurs sont liées à la position X et Y du rectangle.
- La structure utilisée doit être déclarée dans *Control Center → Type de données→ Types de structure* et se présente comme suit :

  Création de variables structurées dans la gestion des variables



4.8 Global Scripts

Dans notre projet, vous pouvez accéder aux exemples sur les *Global Scripts* en sélectionnant le bouton *GSC*. Le bouton droit de la souris D permet d'afficher le code source de l'exemple.

Vue de Global Script

Un clic de la souris sur le bouton *GSC* permet d'afficher la vue ci-dessus. Les différents boutons de la figure correspondent aux exemples décrits. Le bouton droit de la souris D permet d'afficher le code source de l'exemple.

Les fonctions de projet sont créées dans l'éditeur *Global Script* et utilisées dans les actions liées à des objets dans *Graphics Designer*. Les fonctions sont sélectionnées dans l'arborescence des fonctions dans l'éditeur de l'*ActionC*.

4.8.1 Exemple 1 - Utilisation d'une fonction de projet

Nous montrerons dans l'exemple 1 l'utilisation de fonctions de projet dans une action C.
L'action est configurée sur le bouton *exemple_01* → *Evénement* → *Souris* → *Cliquez bouton gauche*.
Lorsque le bouton *Exemple 1* est cliqué avec la souris, le script ci-dessous est édité.

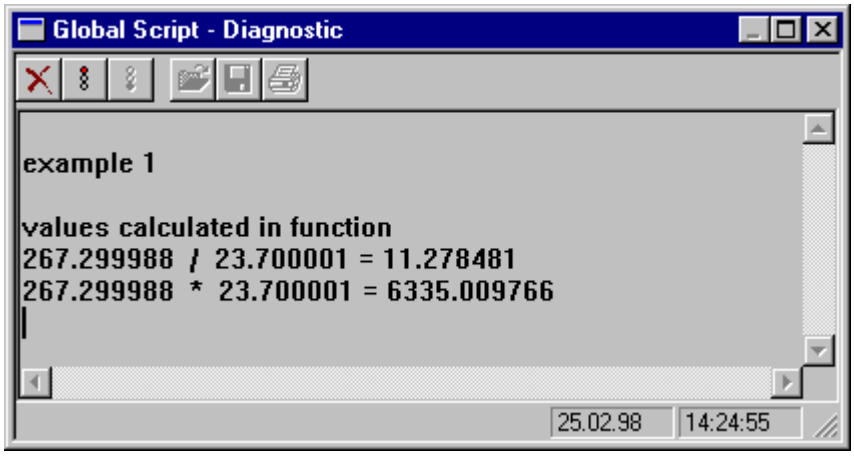
Programme de la fonction de projet

```
void calculate(float a,float b,float *pRes1,float *pRes2)
{
//calculate and store value in addresses
*pRes1=a/b;    //divide
*pRes2=a*b;    //multiply
}
```

Programme de l'action sur l'objet

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
float number1,number2;
float result1,result2;

number1=267.3;
number2=23.7;

//execute function and set parameters
calculate(number1,number2,&result1,&result2);

//output in diagnostics window
printf("\r\nexample 1\r\n");
printf("\r\nvalues calculated in function\r\n");
printf("%f  /  %f = %f\r\n",number1,number2,result1);
printf("%f  *  %f = %f\r\n",number1,number2,result2);
}
```

Affichage dans la fenêtre de diagnostic :



Explication des différentes zones de programme

- Les variables (*numéro1, numéro2*) sont déclarées et la fonction de projet *calculer* est passée dans le programme de l'action.
- Dans le programme de la fonction de projet appelée, les valeurs passées sont calculées. Le passage en retour des valeurs calculées s'effectue par les pointeurs **pRes1, *pRes2*
- Les valeurs de l'*action C* sont ensuite affichées dans la fenêtre de diagnostic par la fonction *printf*.

4.8.2 Exemple 2 - Utilisation de fonctions de projet, autres exemples

Nous montrerons dans l'exemple 2 l'utilisation de fonctions de projet dans une action C.
L'action est configurée sur le bouton *exemple_02* → *Evénement* → *Souris* → *Cliquez bouton gauche* .
Lorsque le bouton *Exemple 2* est cliqué avec la souris, le script ci-dessous est édité.

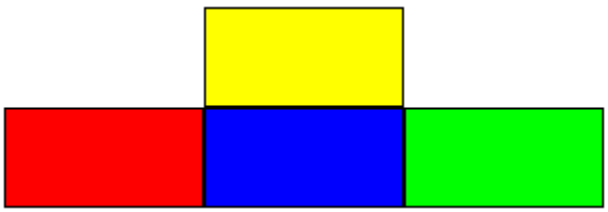
Programme de la fonction de projet

```
void colourchange()
{
long int color1,color2,color3,color4;
long int store;

//get rectangle colours
color1=GetBackColor(lpszPictureName,"rectangle_01");
color2=GetBackColor(lpszPictureName,"rectangle_02");
color3=GetBackColor(lpszPictureName,"rectangle_03");
color4=GetBackColor(lpszPictureName,"rectangle_04");

//cyclic colour change
store=color1;
color1=color2;
color2=color3;
color3=color4;
color4=store;

//set rectangle colours
SetBackColor(lpszPictureName,"rectangle_01",color1);
SetBackColor(lpszPictureName,"rectangle_02",color2);
SetBackColor(lpszPictureName,"rectangle_03",color3);
SetBackColor(lpszPictureName,"rectangle_04",color4);
}
```

Programme de l'action sur l'objet

```
#include "apdefap.h"
void OnLButtonDown(char* lpszPictureName, char* lpszObjectName,
                   char* lpszPropertyName, UINT nFlags, int x, int y)
{
//execute function
colourchange(lpszPictureName);
}
```

Modification des couleurs sur clic de souris :



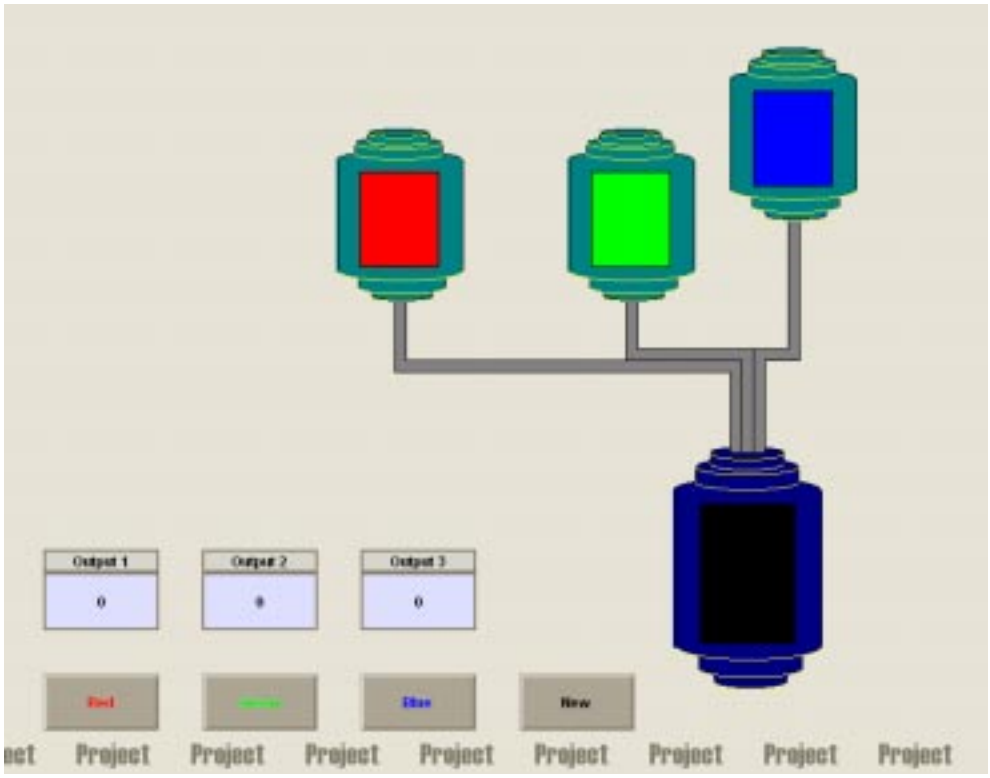
Explication des différentes zones de programme

- Dans le programme de la fonction de projet, les couleurs courantes sont lues, puis déplacées d'une position et réaffectées aux objets.
- A l'appel de la fonction de projet, le nom de vue est passé.

4.9 Exemple de projet

Notre projet est un exemple d'utilisation des thèmes décrits précédemment. Les éléments du programme ne sont pas présentés en détail.

Vue du projet



Lorsque vous avez sélectionné la vue avec le projet, la vue ci-dessus est affichée.
Celle-ci représente une installation de mélange de couleurs. Lorsque vous *cliquez* sur les boutons *rouge*, *vert* et *bleu* avec la , la couleur correspondante est ajoutée avec dosage. Un nouveau *clic* termine le dosage de couleurs ajoutées. Le bouton *Redémarrer* ramène l'installation à son état initial. La quantité déjà déversée (50 maxi) est affichée dans les champs de sortie.

4.10 Affichage du code source avec le bouton droit de la souris

Dans le projet *Cours_00*, les sources de l'*action C* peuvent, pour tous les exemples, être directement affichées en runtime. L'affichage est déclenché, pour tous les exemples, en appuyant sur **D**.

```
#include "apdefap.h"
void OnRButtonDown(char* lpszPictureName, char* lpszObjectName,
                char* lpszPropertyName, UINT nFlags, int x, int y)
{
char name[30],namex[30];
int visx,comp;
int number;
int  ch = '_';
char *pdest;

visx=GetVisible("course_0_startpicture_00.PDL","code picture");
strcpy(namex,GetTagChar("T16x_org_picname_02"));

pdest = strrchr( lpszObjectName, ch );
if ( pdest == NULL )(printf("ObjectNameError"));
else {
   number = atoi(strrchr(lpszObjectName,'_')+1);
   sprintf(name,"course_4_tag_%02d.PDL",number);
   SetTagChar("T16x_org_picname_02",name);
   }
comp=strcmp(name,namex);
if ((comp==0)&&(visx==1)){
      SetVisible("course_0_startpicture_00.PDL","code picture",0);
      SetVisible("course_0_startpicture_00.PDL","code picture",1);
      }
else SetVisible("course_0_startpicture_00.PDL","code picture",1);
}
```

Explication des différentes zones de programme

- La première section est l'*en-tête de fonction*. Elle ne peut être modifiée.
- Déclaration de variable.
- Vérifiez si la fenêtre de vue est déjà sélectionnée (visible).
- Lire le nom de vue.
- Définir le numéro du bouton.
- Composer le nom de la vue de code source.
- Vérification du changement de nom de vue.
- En cas de noms identiques, fermer la vue et la réafficher pour la faire apparaître au premier plan.
- Ceci est affiché s'il s'agit d'une nouvelle vue.

5 Annexe

Vous trouverez dans l'annexe une collection des thèmes qui n'ont pas été intégrés directement dans le *manuel de configuration.*

5.1.1 Entrée/Sortie normalisées dans champ E/S

Les actions ci-dessous doivent être configurées pour qu'un champ E/S sorte l'affichage normalisé d'une valeur ou pour que la valeur entrée soit transmise sous forme normalisée à l'automate :

Action pour Property "Valeur de sortie" d'un champ E/S (important: "float", si des décimales sont souhaitées)

```
float  a;
a=GetTagFloat("DB21_DW1");
return(a/100);
```

Action pour l'événement "Valeur entrée" d'un champ E/S (la variable "Var1" est une valeur 16 bits non signée)

```
float  a;
a=GetInputValueDouble(lpszPictureName,lpszObjectName);
SetTagFloat("Var1",a*100);
```

5.1.2 Actions spécifiques à des objets à la sélection de vue

Il existe des applications dans lesquelles des actions créées pour une propriété d'un ou de plusieurs objets dans une vue ne sont effectuées qu'une seule fois à la sélection de la vue. Une possibilité consiste à formuler une action spécifique des vues pour l'objet vue pour *Evénements→Divers→ Sélection vue*. Ceci présente cependant des inconvénients, car l'action doit agir sur des objets dans la vue et, de ce fait, les objets doivent être nommés de manière fixe dans l'action. Les objets ne sont plus manipulables librement. Cette solution n'est pas orientée objet.

Il est possible de remédier à ce problème :

- Définissez une variable interne (p. ex. *fantôme*) qui ne sera jamais ni rafraîchie ni activée de manière ciblée. Définissez le déclenchement de l'action sur l'objet en fonction d'une modification de cette variable. A la formation de la vue au runtime, l'action est tout d'abord activée et réagirait en principe uniquement sur modification de la valeur de la variable *fantôme*, ce qui ne se produit cependant pas, puisque cette variable n'est jamais modifiée.

5.1.3 WinCC-Scope

Généralités

WinCC-Scope est un utilitaire qui facilite le diagnostic des projets WinCC. Il fournit de nombreuses informations sur le projet activé mais aussi sur l'ordinateur utilisé. Pour pouvoir travailler avec *Scope*, vous devez avoir installé un navigateur tel que Internet Explorer. De même que vous devez avoir intallé le protocole de réseau TCP/IP.

Démarrage et utilisation

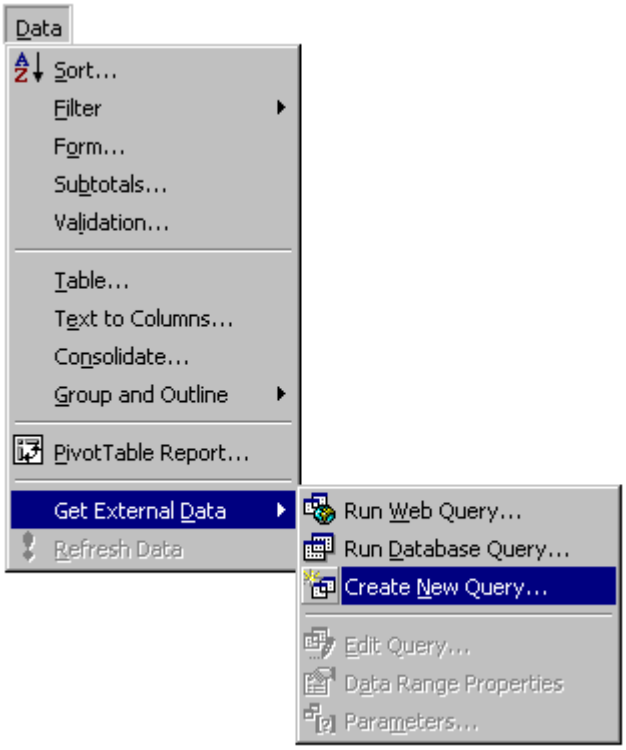
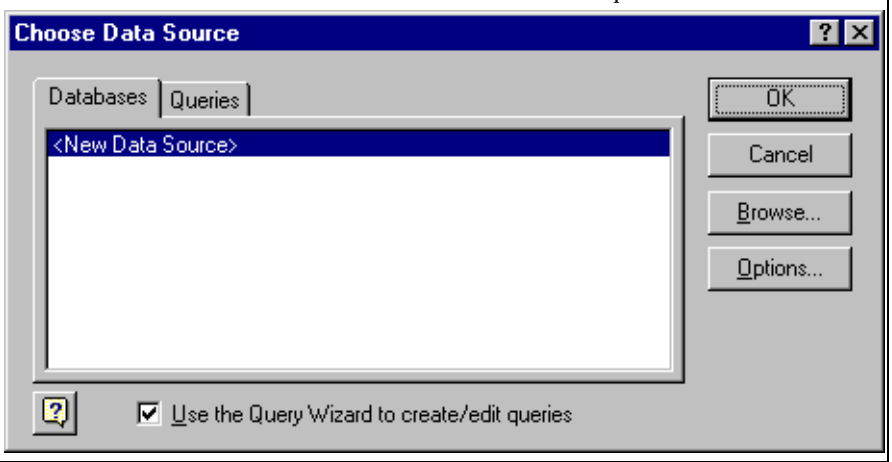
Lors de l'installation de WinCC, *Scope* est également installé par défaut. Avant de pouvoir utiliser Scope, vous devez démarrer le programme *WinCCDiagAgent.exe*. Celui-ci se trouve dans le répertoire *Siemens\WinCC\WinCCScope\bin*. Il s'agit d'un simple serveur HTTP. Vous pourrez ensuite activer *Scope* via le menu de démarrage. La page d'accueil donne accès via le titre *How to use the new Diagnostics Interface* à une page de description générale de l'utilisation de *WinCC-Scope*. Cliquez sur le lien *http://localhost*, pour démarrer *Scope*. La liste dans le volet gauche de la fenêtre fournit des informations sur divers sujets. La rubrique *Info système* contient des informations générales sur l'ordinateur utilisé et la rubrique *Info WinCC* des informations sur le projet WinCC actuellement activé.

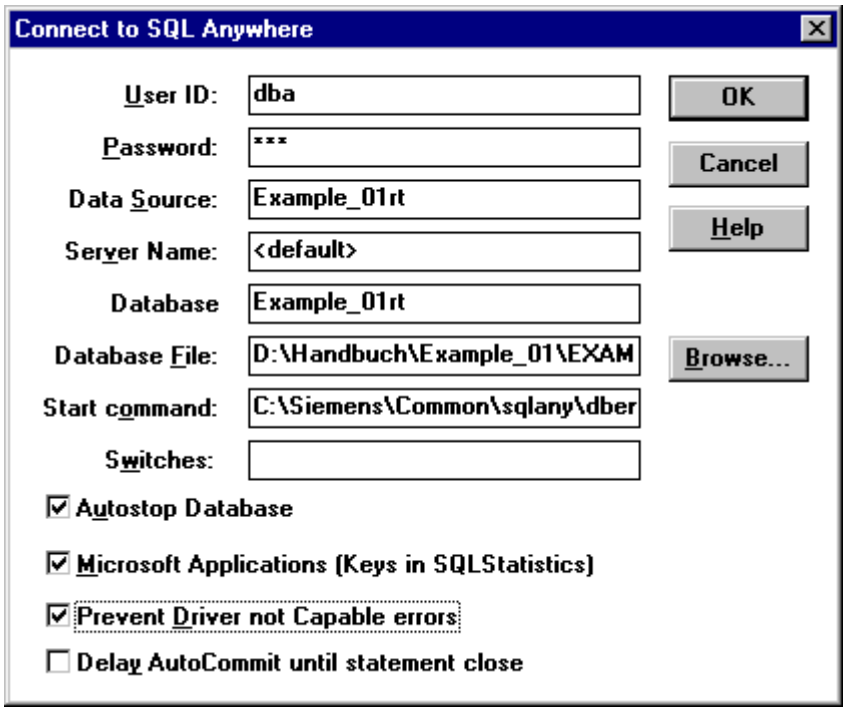
5.1.4 Accès à la base de données

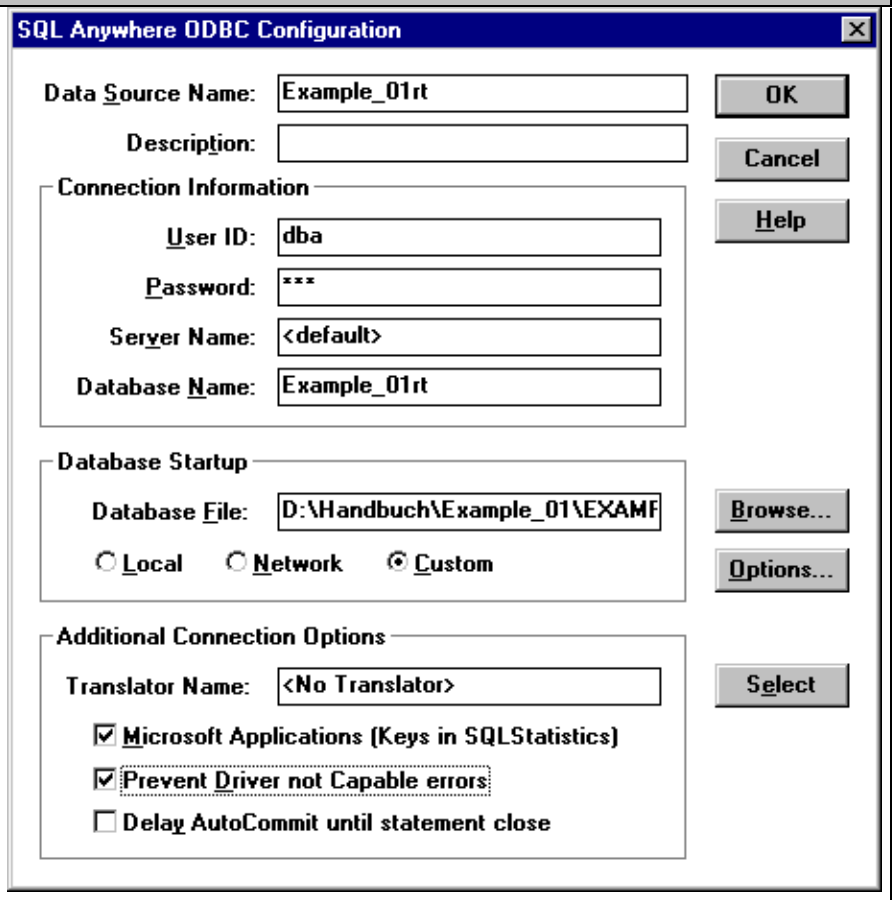
5.1.4.1 Accès à la base de données avec Excel/MSQuery

La description ci-après de l'accès à la base de données WinCC se rapporte à l'utilisation de Microsoft® Excel 97 avec SR-1.

Accès avec Excel/MSQuery

Etape	Procédure: Accès avec Excel/MSQuery
1	Ouvrez Excel. La commande de menu *Données → Données externes → Nouvelle requête...* ouvre le dialogue *Sélectionner source de données* de MSQery.  Dans l'onglet *Bases de données* sélectionnez l'option *<Nouvelle source de données>*. Créerz une nouvelle source de données en cliquant sur le bouton *OK*. 
2	Dans la boîte de dialogue *Créer nouvelle source de données*, entrez le nom de la

Etape	Procédure: Accès avec Excel/MSQuery
	nouvelle source de données. Celui-ci ne doit pas être forcément identique au nom de la base de données WinCC. Sélectionnez comme pilote *Sybase SQL Anywhere 5.0*. Le bouton *Lier...* ouvre le dialogue *Connect to SQL Anywhere* servant à entrer les informations requises par le pilote. Entrez comme *User ID dba* et comme *mot de passe sql*. Le bouton *Browse* permet de sélectionner la base de données à traiter. Validez les entrées par *OK*. 
3	Si aucune source de données n'a encore été configurée pour la base de données sélectionnée, le message *Nom de la source de données introuvable ou pilote standard non spécifié* s'affiche. Validez le message et actionnez à nouveau le bouton *Lier...*. Dans le dialogue *Sélectionner source de données* ouvrez l'onglet *Source de données ordinateur*. Les bases de données CS et runtime du projet WinCC en cours figurent déjà dans la liste des sources de données. Le nom de ces sources de données commence par les caractères *CC_* suivis du nom de projet. Le nom de la source de données représentant la base de données runtime se termine par le caractère *R*. Si vous voulez toutefois traiter une base de données WinCC quelconque, vous devrez d'abord en définir la source de données. Utilisez pour ce faire le bouton *Nouvelle*. Dans l'assistant *Créer une nouvelle source de données* qui s'ouvre, sélectionner à la première page le point *Source de données utilisateur* et passez à la page suivante avec *Suivant*. A la page suivante, sélectionnez le pilote *Sybase SQL Anywhere 5.0* et continuez avec *Suivant*. Fermez la dernière page par *Terminer*. Le dialogue *SQL Anywhere ODBC Configuration* servant à entrer les informations requises par le pilote, s'ouvre. Entrez à nouveau comme *User ID dba* et comme *mot de passe sql*. Le bouton *Browse* permet de sélectionner la base de données à traiter. Fermer le dialogue avec *OK*.

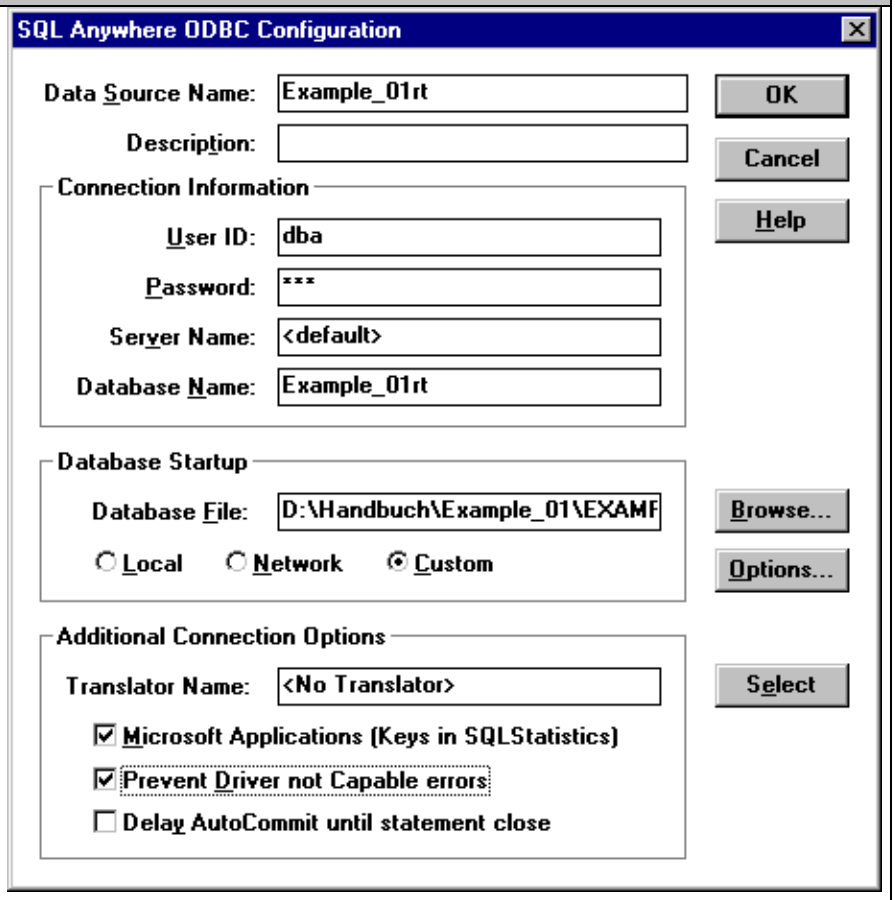
Etape	Procédure: Accès avec Excel/MSQuery
	 Sélectionnez la source de données qui vient d'être créée dans le dialogue *Sélectionner source de données* puis validez le dialogue par *OK*. Validez le dialogue *Connect to SQL Anywhere* qui s'affiche à présent. La source de données peut être préalablement configurée à l'aide du panneau de configuration. Ouvrez, dans le panneau de configuration, l'*administrateur de sources de données ODBC*. Le bouton *Ajouter* permet également de démarrer l'assistant *Créer une nouvelle source de données*.  ODBC
4	Validez le dialogue *Créer une nouvelle source de données* par *OK*. Dans le dialogue *Sélectionner source de données* sélectionnez la source de données qui vient d'être créée puis validez le dialogue par *OK*. La première page de l'*assistant Query* qui s'ouvre, affiche à présent toutes les feuilles de calcul et colonnes disponibles. Sélectionnez les feuilles de calcul et les colonnes voulues puis validez votre choix par *Suivant*. La page suivante permet de définir les filtres ainsi que l'odre de tri des données. La dernière page sert à indiquer si les données seront traitées par la suite sous Excel ou MSQuery. Fermez le dialogue par *Terminer*.
5	Le dialogue *Renvoyer les données externes vers Excel* qui s'ouvre, permet de définir la position des feuilles de calcul à insérer. Vous pouvez en outre y définir les propriétés de la zone de données externe. Quitter le dialogue avec *OK*.

5.1.4.2 Accès à la base de données avec Access

La description ci-après de l'accès à la base de données WinCC se rapporte à l'utilisation de Microsoft® Access 97 avec SR-1.

Accès avec Access

Etape	Procédure: Accès avec Access
1	Ouvrir ou créer une base de données Access. La commande de menu *Fichier → Données externes → Importer...* ouvre le dialogue *Importer*. Sélectionnez comme *type de fichier* l'option *Bases de données ODBC()*. Le dialogue *Sélectionner source de données* s'ouvre automatiquement. Dans l'onglet *Source de données ordinateur* sélectionnez une source de données. Les bases de données CS et runtime du projet WinCC en cours figurent déjà dans la liste des sources de données. Le nom de ces sources de données commence par les caractères *CC_* suivis du nom de projet. Le nom de la source de données représentant la base de données runtime se termine par le caractère *R*.
2	Si la base de données WinCC ne figure pas encore dans la liste, il convient de créer d'abord une source de données à l'aide du bouton *Nouvelle*. Dans l'assistant *Créer une nouvelle source de données* qui s'ouvre, sélectionnez à la première page le point *Source de données utilisateur* et passez à la page suivante avec *Suivant*. A la page suivante, sélectionnez le pilote *Sybase SQL Anywhere 5.0* et continuez avec *Suivant*. Fermez la dernière page par *Terminer*. Le dialogue *SQL Anywhere ODBC Configuration* servant à entrer les informations requises par le pilote, s'ouvre. Entrez comme *User ID dba* et comme *mot de passe sql*. Le bouton *Browse* permet de sélectionner la base de données à traiter. Fermer le dialogue avec *OK*.

Etape	Procédure: Accès avec Access
	 Sélectionnez la source de données qui vient d'être créée dans le dialogue *Sélectionner source de données* puis validez le dialogue par *OK*.
3	Dans le dialogue *Importer objets* qui s'ouvre, vous pouvez sélectionner les tableaux de base de données voulus. *OK* permet de les insérer dans la base de données Access.

5.1.4.3 Accès à la base de données avec ISQL

Le langage ISQL permet d'accéder directement à la base de données WinCC. Ceci est effectué par l'utilisateur sous sa propre responsabilité, l'édition ou la suppression de tables pouvant entraîner la perte d'intégrité du référentiel des données de configuration.

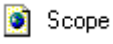
Accès avec ISQL

Etape	Procédure: Accès avec ISQL
1	Démarrez ISQL.EXE dans le répertoire Siemens\Common\sqlany. Le dialogue *Interactive SQL Logon* s'ouvre. Entrez comme *User ID dba* et comme *mot de passe sql*. Si vous validez par OK, le programme est automatiquement lié à la base de données WinCC ouverte actuellement, à savoir à la base de données CS. Si vous voulez toutefois accéder à une autre base de données, p. ex. à la base de données runtime, sélectionnez la commande de menu *Command → Connect*. Dans le dialogue qui s'ouvre entrez à nouveau *User ID* et *mot de passe*. Comme *Database File*, indiquez la base de données voulue avec

Etape	Procédure: Accès avec ISQL
	sont chemin complet.
2	Vous pouvez à présent entrer dans la fenêtre Command des instructions SQL qui seront exécutées après actionnement du bouton *Execute*. Voici quelques exemples d'instructions SQL : • select * from systable : affiche tous les noms de tableau • select * from <*nom de tableau*> : affiche le contenu du tableau appelé <*nom de tableau*> • unload tabele <*nom de tableau*> to <*nom de fichier*> : exporte le tableau appelé <*nom de tableau*> dans le fichier appelé <*nom de fichier*> • drop table <*nom de tableau*> : supprime le tableau appelé <*nom de tableau*>

5.1.4.4 Accès à la base de données avec WinCC-Scope

Accès avec WinCC-Scope

Etape	Procédure: Accès avec WinCC-Scope
1	Avant de démarrer WinCC-Scope dans le menu de démarrage, il convient de démarrer l'application WinCCDiagAgent.exe dans le dossier *Siemens\WinCC\WinCCScope\bin*. 
2	La page d'accueil donne accès via le titre *How to use the new Diagnostics Interface* à une description générale de l'utilisation de WinCC-Scope. Cliquez sur le lien *http://localhost*, pour démarrer Scope.
3	Dans la zone gauche, une liste propose diverses fonctions. • L'option *Database* donne accès à des informations générales sur la base de données WinCC. • L'option *Database Query* permet d'afficher les tableaux d'une base de données. La *Data Source* par défaut est la base de données CS du projet WinCC actuellement ouvert. Le nom de ces sources de données commence par les caractères *CC_* suivis du nom de projet. Le nom de la source de données représentant la base de données runtime se termine par le caractère *R*. Il est toutefois possible d'afficher également d'autre sources de données. • L'option *SQL Query* permet d'appliquer des instructions SQL à une source de données sélectionnée. Il est cependant conseillé de ne traiter la base de données WinCC avec des instructions SQL qu'avec d'excellentes connaissances du système. Vous trouverez des exemples d'instructions SQL à la section précédente intitulée *Accès à la base de données avec ISQL*.

5.1.4.5 Exportation à partir de la base de données avec des actions C

L'exportation de données peut également être activée depuis une vue de WinCC runtime. On peut en outre lancer un SQL Interactive avec une ligne de commande via ProgramExcecute. L'action à exécuter est rangée dans un fichier de commandes (dans l'exemple: archiv.sql).

Action C p. ex. sur bouton

```
#include "apdefap.h"
void OnClick(char* lpszPictureName, char* lpszObjectName, char*
                lpszPropertyName)
{

char* path =   "C:\\SIEMENS\\Common\\SQLANY\\ISQL -q -b -c";
char* parameters =
              "UID=DBA;PWD=SQL;DBN=CC_Project_97-10-21_09:53:27R";
char* action = "read D:\\WinCC\\Project\\archiv.sql";

char ExportArchive[200];

sprintf(ExportArchive,"%s %s %s",path,parameters,action);

ProgramExecute(ExportArchive);
}
```

- La variable *path* contient le chemin du programme ISQL.exe ainsi que ses paramètres d'appel.
- La variable *parameters* contient les informations entrées dans le dialogue *Interactive SQL Logon* pour la connexion à la base de données. Ces informations sont :
  - UID (User ID) : DBA
  - PWD (Password) : SQL
  - DBN (Data Base Name) : Nom de la source de données ODBC. Le nom de ces sources de données commence par les caractères *CC_* suivi du nom de projet, de la date et de l'heure de la création du projet. Le nom de la source de données représentant la base de données runtime se termine par le caractère *R*. Lorsque le projet est activé, ce nom peut être déterminé à l'aide de *Panneau de configuration → ODBC → onglet DSN utilisateur*.
- La variable *action* spécifie l'exécution des instructions SQL se trouvant dans le fichier *archiv.sql*.
- Les instructions sont regroupées dans *ExportArchive* et exécutées par la fonction *ProgramExecute()*.

Nota:
Si vous voulez exporter à partir d'une base de données autre que l'une des deux bases de données du projet, spécifiez au lieu du paramètre DBN le paramètre DBF, c.-à-d. le fichier de base de données avec le chemin de la base de données. Cette possibilité ne s'applique cependant pas à la base de données de projet activée.

Contenu du fichier: archiv.sql

```
select * from PDE#HD#ProcessValueArchive#Analog;
output to D:\WinCC\Projekt\archiv.txt format ascii
```

- Dans la base de données ouverte, l'archive de valeurs de mesure *pde#hd#ProcessValueArchive#analog* est sélectionnée et exportée dans le fichier *ASCII archiv.txt* à l'aide de la commande output.

5.1.4.6 Sélection de tables dans la base de données

La commande *select* décrite ci-dessus du fichier de commandes sélectionne les tables. L'utilisation de paramètres supplémentaires permet de sélectionner des sous-ensembles de cette table qui seront alors exportés avec la commande *output*. Voici quelques exemples à ce sujet.

Sélection dans une période

```
select * from PDE#HD#ProcessValueArchive#Analog where T between
                ’1996-5-1 10:10:0.00’ and ’1996-6-1 10:10:0.00’
```

Sélection à partir d'un instant

```
select * from PDE#HD#ProcessValueArchive#Analog where T >
                ’1996-5-1 10:10:0.00’
```

Sélection de valeurs de process avec et sans tri

```
select * from PDE#HD#ProcessValueArchive#Analog where V > 100
select * from PDE#HD#ProcessValueArchive#Analog where V > 100
                order by T
```

Sélection de valeurs de process avec choix des colonnes T (Time) et V (Value)

```
select T,V from PDE#HD#ProcessValueArchive#Analog where V > 100
                order by T
```

5.1.5 Couplage série

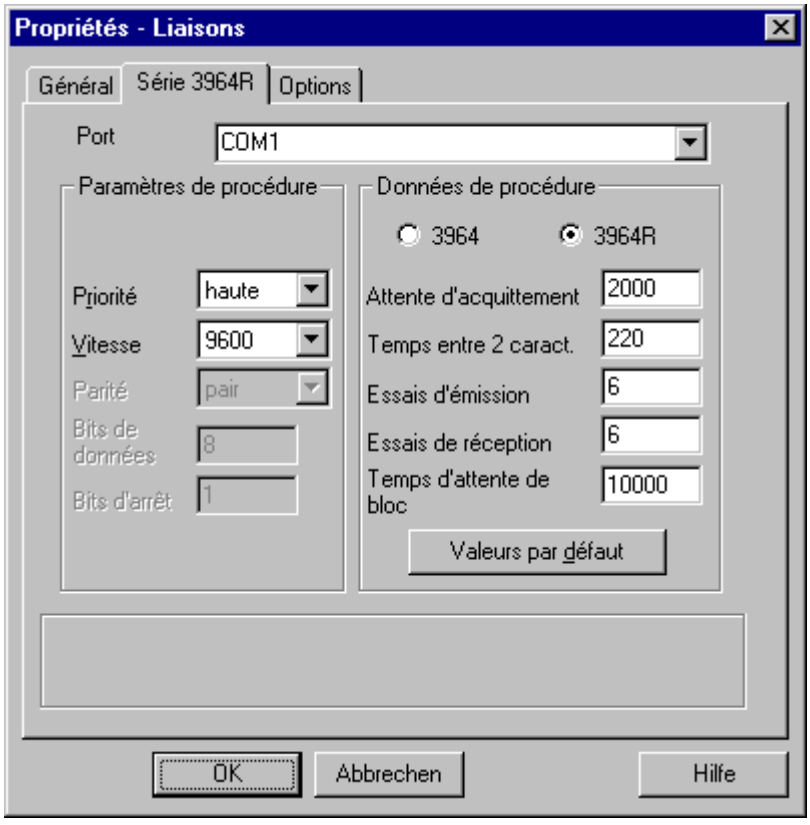
Les paramétrages ci-dessous sont nécessaires pour établir une liaison série:

Paramétrages CP525:

Message:	Paramètre CP525 Nom:	P3964R
Procédure:	COMPOSANT: RK	Version : 01
Vitesse::	9600	Bits de données : 8
Bits d'arrêt: 1	Priorité: basse	
Parité:	PAIR	

Dans l'automate correspondant, paramétrez une fois SYNCHRON dans la branche de démarrage pour le CP525 et SEND/RECEIVE-ALL dans le programme cyclique.

Paramétrages WinCC :



Pour optimiser la liaison, un des deux partenaires doit avoir la priorité *haute*, logiquement WinCC.

5.1.6 Table des couleurs

Les valeurs de couleur sont composées à partir d'une grande palette de couleurs.
Les 16 couleurs de base sont:

Couleur	Valeur de couleur (Hex)	Constante symbolique
Rouge	0x000000FF	CO_RED
Rouge foncé	0x00000080	CO_DKRED
Vert	0x0000FF00	CO_GREEN
Vert foncé	0x00008000	CO_DKGREEN
Bleu	0x00FF0000	CO_BLUE
Bleu foncé	0x00800000	CO_DKBLUE
Cyan	0x00FFFF00	CO_CYAN
Cyn foncé	0x00808000	CO_DKCYAN
Jaune	0x0000FFFF	CO_YELLOW
Jaune foncé	0x00008080	CO_DKYELLOW
Magenta	0x00FF00FF	CO_MAGENTA
Magenta foncé	0x00800080	CO_DKMAGENTA
Gris clair	0x00C0C0C0	CO_LTGRAY
Gris	0x00808080	CO_DKGRAY
Noir	0x00000000	CO_BLACK
Blanc	0x00FFFFFF	CO_WHITE

Constante symbolique prédifinie avec #define extern.
Les couleurs intermédiaires s'obtiennent avec des valeurs intermédiaires de la palette.

Lorsque des modifications de valeurs sont générées à l'aide du dialogue de dynamisation et si les données configurées sont ensuite traitées avec des actions C, il est également possible de lire les valeurs de couleur ; celles-ci sont cependant au format décimal.

5.2 Documentation du système d'alarmes S5

Tâche et fonctionnement du système d'alarmes S5

La présente documentation décrit les fonctions et les propriétés du logiciel SIMATIC S5 :

Logiciel Système d'alarmes S5
Ce logiciel sert à l'acquisition chronologique d'alarmes binaires ainsi qu'à leur traitement et leur stockage en tampon. Dans l'environnement SIMATIC S5, ce package fournit les fonctionnalités logicielles nécessaires à la réalisation de la fonction 'Acquisition chronologique d'alarmes' du système WinCC.

Le principe de fonctionnement du logiciel peut être présenté ainsi : Le logiciel surveille les états de signaux binaires des alarmes qui sont mis par l'utilisateur à la disposition du système d'alarmes S5 dans des DB interface. En présence d'un changement d'état d'un signal, l'alarme correspondante est identifiée à l'aide de son numéro et horodatée. Ces données (à condition d'avoir été configurées par l'utilisateur) sont complétées par une variable de process 32 bits et une désignation alphanumérique d'ordre de fabrication / charge. Si nécessaire, le bloc d'alarme ainsi formé est stocké dans un tampon cyclique. Un stockage en tampon des données d'alarme est toujours nécessaire lorsque plus d'alarmes surviennent dans l'unité de temps que le système ne peut transmettre à WinCC par le bus. Cette fonctionnalité permet un découplage temporel entre l'acquisition chronologique des alarmes dans SIMATIC S5 et le système d'alarmes WinCC ainsi qu'un traitement des alarmes en temps réel.

Les blocs d'alarme créés par le système d'alarmes S5 sont mis à la disposition du programme utilisateur S5 dans une interface. Une connexion logicielle S5 (à réaliser par l'utilisateur) transmet ces données par une liaison bus (p. ex. SINEC H1) au système d'alarmes WinCC. Celui-ci dispose de fonctions étendues de traitement des alarmes, telles que visualisation, archivage, consigne dans des journaux, etc.

La configuration du système d'alarmes S5 par l'utilisateur s'effectue dans un DB interface (DB système 80). Dans cette interface, l'utilisateur définit le cadre système dans lequel fonctionnera le système d'alarmes. C'est ici que se définissent les zones mémoire utilisées par le système d'alarmes S5, le type et la taille des alarmes à traiter, ainsi que la répartition des zones d'adressage occupées.

Le présent chapitre décrit l'utilisation et la manipulation des systèmes d'alarmes S5 dans l'environnement SIMATIC S5. Une vue d'ensemble des fonctions et des blocs de données utilisés par le logiciel ainsi que des espaces mémoire nécessaires est présentée à l'utilisateur. Vient ensuite une description détaillée de toutes les interfaces de données disponibles entre le système d'alarmes S5 et le programme utilisateur S5. Un exemple de configuration est destiné à faciliter à l'utilisateur la prise en main du système d'alarmes S5.

5.2.1 Liste des blocs logiciels

Le logiciel SIMATIC S5 fourni se trouve sur le CD ROM, avec les exemples relatifs au présent manuel, dans le fichier *'WINCC1ST.S5D'*.
Ce fichier contient les blocs fonctionnels et blocs de données ci-dessous pour le système d'alarmes S5 :

FB	Nom	Taille	Fonction
FB 80	SYSTEMFB	1114	Acquisition chronologique des alarmes
FB 81	ANLAUFFB	135	Lancement et initialisation de l'acquisition chronologique des alarmes
FB82	PCHECK	574	Appelé par FB 81
FB 83	MBLOCK	699	Appelé par FB 80
FB 84	SCHREIB	94	Appelé par FB 80
FB 87	VOLL	87	Appelé par FB 80
DB 80	DB Système	512	Paramétrage du système d'alarmes
Total		2703	

Tableau 1

L'espace mémoire minimum nécessaire dépend de la configuration du système d'alarmes S5. Les blocs de données ci-dessous sont toujours également nécessaires.

Tampon cyclique (mini) *2 DB = 1024 octets*
Interface transfert à WinCC *1 DB = 512 octets*

Comptez en outre 512 autres octets par DB d'offset et de paramètres.

Le calcul de la taille exacte des blocs de données d'offset est décrit au chapitre 5.2.4.1 Structure du bloc de données d'offset, de même que vous trouverez au chapitre 5.2.4.10 Structure du bloc de données de paramètres - le calcul de la taille des blocs de données de paramètres.

5.2.2 Conditions à remplir par le matériel

Les blocs fonctionnels indiqués dans le Tableau 1 pour le système d'alarmes S5 nécessitent, pour l'exécution correcte des fonctions, le matériel suivant :

AG	CPU
AG 115U	CPU 944 * , CPU 945
AG 135U	CPU 928B
AG 155U	CPU 946/ 947, CPU 948

Tableau 2
* seule la CPU 944 dotée de deux interfaces PG possède une heure système

Ces CPU disposent d'une heure interne et peuvent par conséquent fournir la date et l'heure courantes pour la formation des blocs d'alarme.
Un télégramme d'horodatage courant est écrit cycliquement, pour chaque canal WinCC créé, dans la CPU SIMATIC S5. Le bloc fonctionnel **FB 86 : MELD:UHR** synchronise l'heure interne de l'automate SIMATIC S5 avec l'heure du système WinCC.

5.2.3 Intégration du système d'alarmes S5 dans le programme utilisateur SIMATIC-S5

Les opérations suivantes sont nécessaires pour intégrer le logiciel du système d'alarmes SIMATIC S5 dans le programme utilisateur SIMATIC S5 :

Tous les blocs de données figurant dans Tableau 1 peuvent être transférés dans l'automate par le fichier **WinCCST.S5D**.
S'ils n'ont pas déjà été implémentés en standard ou ne sont pas encore installés sur l'automate programmable, les blocs de manipulation doivent être transférés dans l'automate correspondant.

Opération	Procédure: Intégrer le système d'alarmes
1	Tous les blocs de données figurant dans Tableau 1 peuvent être transférés dans l'automate par le fichier **WinCCST.S5D**.
2	S'ils n'ont pas déjà été implémentés en standard ou ne sont pas encore installés sur l'automate programmable, les blocs de manipulation doivent être transférés dans l'automate correspondant.
3	Paramétrez le bloc de données DB 80 selon le chapitre5.2.6.
4	Créez les blocs de données pour le DB de transfert, le tampon cyclique, les offsets d'alarme et éventuellement les paramètres d'alarme (voir chapitre 5.2.4).
5	Initialisez les blocs de données d'offset pour les différents types d'alarmes (voir chapitre 5.2.4.1- Etats de repos, numéro d'alarme de base,...).
6	Indiquez les variables de process, la désignation d'ordre de fabrication / charge pour les diverses alarmes dans le programme utilisateur (voir chapitre 5.2.4.10).
7	Appelez les blocs ci-dessous dans les OB de lancement (OB 20, OB 21, OB 22) : • **SPA HTB : SYNCHRON** (Le bloc de manipulation de la CPU correspondant) SPA FB 81 : FB de lancement
8	Appelez les blocs suivants dans OB 1 : • pour le traitement cyclique des alarmes **SPA FB 80 : SYSTEMFB** • un bloc fonctionnel créé par l'utilisateur pour la transmission des blocs d'alarme au système WinCC (voir chapitre 5.2.5.4)
9	Ajoutez d'autres fonctionnalités selon le chapitre suivant : • La synchronisation de la date et de l'heure par FB 86 : MELD:UHR (voir chapitre 5.2.8).

Tableau 3

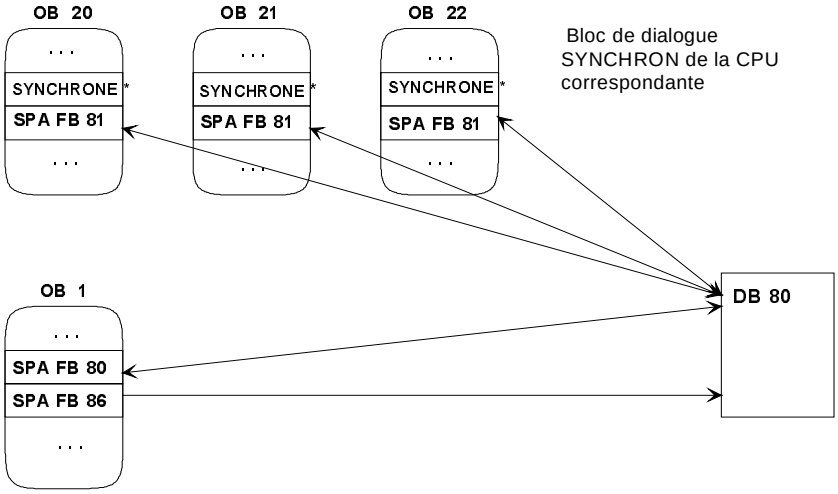

OB 20
SYNCHRONE *
SPA FB 81
OB 21
SYNCHRONE *
SPA FB 81
OB 22
SYNCHRONE *
SPA FB 81
Bloc de dialogue
SYNCHRON de la CPU
correspondante
OB 1
SPA FB 80
SPA FB 86
DB 80

Figure 1

5.2.4 Description générale du système d'alarmes S5s

Ce chapitre décrit les composants du système d'alarmes S5 :

- Bloc de données d'offset
- Bloc de données de paramètres
- Bloc d'alarme
- Tampon cyclique
- Interface de transfert (DB)
- Bloc de données système

Relations entre les différents composants :

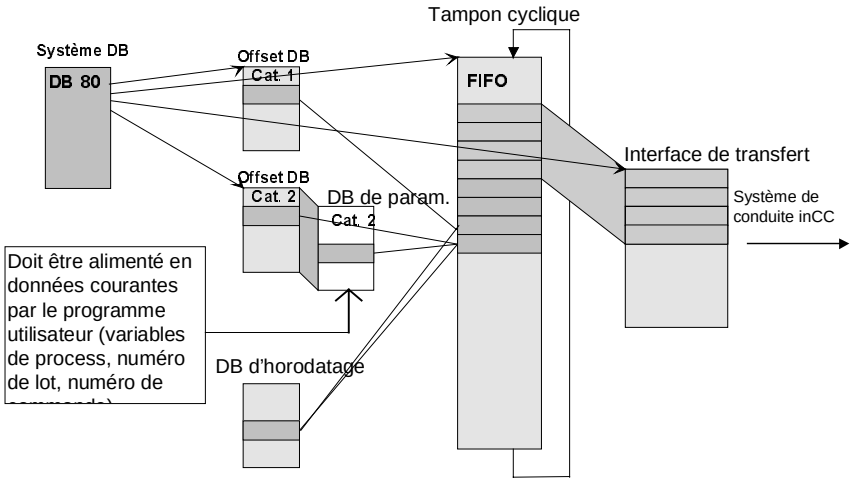


Figure 2

Pour que le système d'acquisition des alarmes puisse surveiller et acquérir les alarmes, celles-ci doivent être configurées dans les blocs de données correspondants. On distingue les quatre types d'alarmes suivants :

Type	Définition : Types d'alarmes
1	Alarme sans paramètres
2	Alarme avec variables de process (2 mots de données)
3	Alarme avec variables de process (2 mots de données) et désignation d'ordre de fabrication/charge (3 mots de données)
4	Alarme avec variables de process (2 mots de données) et désignation d'ordre de fabrication/charge (3 mots de données) et réserve (3 mots de données)

Tableau 4

Un horodatage global peut être affecté pour la formation des blocs d'alarme. En cas d'absence d'horodatage, le système WinCC complète les blocs d'alarme (voir chapitre 5.2.4.11) avec les informations correspondantes.

5.2.4.1 Structure du bloc de données d'offset

La structure du bloc de données d'offset est la même pour les quatre types d'alarmes. L'adresse de bloc de données est indiquée pour chaque type d'alarme nécessaire (voir 5.2.4.1) dans le bloc de données système DB 80.

Bloc de données d'offset pour le type d'alarme correspondant :

DW	Contenu	Affectation
DW 0	sans affectation	En-tête
DW 1	Numéro d'alarme de base	
DW 2	Adresse du dernier bloc d'états de signaux	
DW 3	sans affectation	
DW 4	Etats des signaux des alarmes - bit no : 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Bloc d'état de signaux 1
DW 5	Bits d'état de repos	
DW 6	Bits d'acquittement	
DW 7	Mémentos de front de signaux	
DW 8	Etats des signaux des alarmes - bit no : 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Bloc d'état de signaux 2
DW 9	Bits d'état de repos	
DW 10	Bits d'acquittement	
DW 11	Mémentos de front de signaux	
DW 12	Etats des signaux des alarmes - bit no : 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0	Bloc d'état de signaux 3
DW 13	Bits d'état de repos	

Tableau 5

Nous décrivons ci-dessous :

- Numéro d'alarme de base
- Numéro d'offset
- Bloc d'états de signaux
- Adresse du dernier bloc d'états de signaux
- Etats de signaux
- Bits d'état de repos
- Bits d'acquittement
- Mémentos de front de signaux

5.2.4.2 Numéro d'alarme de base

A chaque alarme est affectée un numéro d'alarme permettant d'identifier les alarmes qui surviennent. Ce numéro d'alarme est composé du numéro d'alarme de base et d'un numéro d'offset.
Un numéro d'alarme de base différent doit être indiqué pour chaque type d'alarme utilisé. A partir de ce numéro d'alarme de base, les alarmes de ce type sont distinguées par les numéros d'offset croissants.
Un numéro d'alarme de base correspondant au type d'alarme est indiqué dans le DW 1 du bloc de données d'offset correspondant (voir 5.2.4.10).

Cas particulier

Lorsqu'on utilise le type d'alarme 1, il est possible d'utiliser deux blocs de données d'offset. Pour obtenir une numérotation continue des alarmes de ce type, entrez comme numéro d'alarme de base du second bloc de données d'offset le numéro d'alarme de base du premier bloc de données d'offset plus la capacité de celui-ci (1008 alarmes).

Calcul d'un numéro d'alarme :
Numéro d'alarme = Numéro d'alarme de base + Numéro d'offset

Exemple :

Calcul	Description :
Données :	Type d'alarme 1, numérotation d'alarmes continue, débutant au numéro d'alarme 10000
A calculer :	Numéro d'alarme de base des deux blocs de données d'offset
10000	Numéro d'alarme de base du premier bloc de données d'offset
10000 + 1008 = 11008	Numéro d'alarme de base du deuxième bloc de données d'offset

5.2.4.3 Numéro d'offset / Etats des signaux des alarmes

Les états des signaux des alarmes sont rangés dans les blocs de données d'offset du type d'alarme correspondant, à la position de bit correspondant au numéro d'alarme d'offset.
Le numéro d'offset de l'alarme correspondante est indiqué à partir des 16 bits (bit 0 à 15) du DW 4. La numérotation continue utilise ensuite un pas de '4' (DW8, DW12, ...).

Bloc d'états de signaux	Bloc d'états de signaux commençant au DW	Numéro de bit 0 - 15 correspondant au numéro d'offset
1	4	0 - 15
2	8	16 - 31
3	12	32 - 47
4	16	48 - 63
...		...
62	248	976 - 991
63	252	992 - 1007

Tableau 6

Calcul du numéro d'offset :
Numéro d'offset = numéro d'alarme - numéro d'alarme de base
*Numéro d'offset = (mot de données / 4 -1) * 16 + n° bit (0-15)*
*Numéro d'offset = (bloc d'états de signaux - 1) * 16 + n° bit (0-15)*

Calcul du DB, DW, n° de bit à partir du numéro d'offset :
Bloc de données = bloc de données d'offset
*Mot de données = (numéro d'offset / 16+ 1) * 4*
N° bit = numéro d'offset % 16

Pour le type d'alarme 1, le mot de données en résultant peut être plus grand que 252 ; dans ce cas :
Bloc de données = bloc de données d'offset + 1
Mot de données = mot de données - 252
N° bit = N° bit

Exemple 1 :

Données : *DW 248, bit 7, numéro d'alarme de base = 10000*
A calculer : *Numéro d'alarme*

Bloc d'états de signaux = *248 / 4*
= *62*
Numéro d'offset = *(bloc d'états de signaux - 1) * 16 + n° bit*
= *(62 - 1) * 16 + 7 = 983*
Numéro d'alarme = *Numéro d'alarme de base + Numéro d'offset*
= *10000 + 983 = 10983*

Le numéro d'alarme souhaité est 10983.

Exemple 2 :

Données : *Type d'alarme 1 avec deux blocs de données d'offset, numéro d'alarme = 12000, numéro d'alarme de base = 10000*
A calculer : *DB, DW, n° bit*

Numéro d'offset = *numéro d'alarme - numéro d'alarme de base*
= *12000 - 10000 = 2000*
N° bit = *numéro d'offset % 16 = 0*
Mot de données = *(numéro d'offset / 16+ 1) * 4*
= *(2000 / 16 + 1) * 4 = 504*

Le mot de données est plus grand que 252.

Bloc de données = *bloc de données d'offset + 1*
Mot de données = *504 - 252 = 252*
N° bit = *0*

Le numéro d'alarme 12000 se trouve dans le deuxième bloc de données d'offset du premier type d'alarme, dans le mot de données 252, bit n° 0.

5.2.4.4 Bloc d'états de signaux

Le premier bloc d'états de signaux commence à l'adresse de mot de données 4. Les blocs d'états de signaux suivants sont séparés par 4 mots de données (DW 8, DW 12, ...).
Voir aussi Tableau 5 ou Tableau 6.

63 blocs d'états de signaux sont possibles pour chaque bloc de données d'offset (bloc d'états de signaux 1 à 63).

Un bloc d'états de signaux contient 16 états de signaux. Ceci donne 63 * 16 = 1008 alarmes possibles dans un bloc de données d'offset.

Structure du bloc d'états de signaux :

DW	Numéro de bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	Etats de signaux	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	Etats de repos	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	Bits d'acquittement	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	Mémentos de front de signaux	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tableau 7

D'autres informations sur ces quatre états binaires seront fournies dans la suite de ce chapitre.

Calcul d'un bloc d'états de signaux :
Bloc d'états de signaux = (numéro d'offset / 16) + 1
Bloc d'états de signaux = mot de données / 4

Calcul du mot de données par lequel commence un bloc d'états de signaux :
*Premier mot de données du bloc d'états de signaux = bloc d'états de signaux * 4*

5.2.4.5 Adresse du dernier bloc d'états de signaux

Le nombre d'alarmes possibles pour un type d'alarme est indiqué à l'aide de l'adresse DW, du dernier bloc d'états de signaux occupé par des alarmes.

Calcul du dernier bloc d'états de signaux :

dernier bloc d'états de signaux = alarmes nécessaires de ce type d'alarme / 16

```
// bloc d'états de signaux pas rempli complètement (16 alarmes)
if ((alarmes nécessaires de ce type d'alarme % 16) != 0)
{
      ++ dernier bloc d'alarme de signal;
}
```

Pour le type d'alarme 1, un nombre d'alarmes plus grand que 1008 peut se rencontrer ; dans ce cas :

1er DB d'offset :

dernier bloc d'états de signaux = 63
*Adresse du dernier bloc d'états de signaux = 63 * 4 = 252*

2e DB d'offset :

dernier bloc d'états de signaux = (alarmes nécessaires de ce type d'alarme - 1008) / 16

```
// bloc d'états de signaux pas rempli complètement (16 alarmes)
if (((alarmes nécessaires de ce type d'alarmes - 1008) % 16) != 0)
{
      ++ dernier bloc d'alarme de signal;
}
```

Calcul de l'adresse DW du dernier bloc d'états de signaux :
Adresse DW du
*dernier bloc d'états de signaux = dernier bloc d'états de signaux * 4*

Exemple :

Données : 1030 alarmes du type d'alarme 1

1er DB d'offset :

*Adresse du dernier bloc d'états de signaux = 63 * 4 = 252*

2e DB d'offset :

alarmes nécessaires - 1008 = 1030 - 1008 = 22
(alarmes nécessaires - 1008) / 16 = 22 / 16 = 1
(alarmes nécessaires - 1008) % 16 = 22 % 16 = 6
dernier bloc d'états de signaux = 2
*Adresse du dernier bloc d'états de signaux = 2 * 4 = 8*

5.2.4.6 Etats de signaux

Position : 1er mot de données du blocs d'états de signaux (voir Tableau 5).

L'utilisateur doit entrer les états de signaux des alarmes correspondantes dans les mots de données prévus à cet effet des blocs de données d'offset du type d'alarme correspondant. Ceci peut être réalisé par rafraîchissement continu des signaux en parallèle au déroulement du process par le programme utilisateur de l'automate.

5.2.4.7 Etats de repos

Position : 2e mot de données du bloc d'états de signaux (voir Tableau 5).

On entend par état de repos d'un signal le niveau de ce signal à l'état passif de fonctionnement. On convient ainsi qu'un signal (alarme) sera actif sur les positions 'low' ou 'high'. Cette information est nécessaire pour déterminer si une alarme 'arrive' ou 'part'.
Lorsqu'un changement d'événement possède l'état inversé pour l'état de repos, il s'agit d'une alarme 'arrivant'. Pour une alarme 'partant', l'état du changement d'événement est le même que celui de l'état de repos correspondant.

Les états de repos des alarmes doivent être indiqués par l'utilisateur aux positions correspondantes.

5.2.4.8 Bits d'acquittement

Position : 3e mot de données du bloc d'états de signaux (voir Tableau 5).

Les bits d'acquittement ne sont pas configurés, mais ils sont évalués par le programme en cours d'exécution. Les alarmes sont alors acquittées directement par le PC de niveau supérieur selon la philosophie d'acquittement utilisée. Le PC envoie ces acquittements spécifiques aux alarmes à l'automate programmable concerné avec les alarmes configurées à cet effet du système d'acquisition d'alarmes intégré.

Le bit d'acquittement correspondant est mis à 1 par le système d'alarmes S5 pour un temps de cycle de l'automate.
Le programme utilisateur doit exploiter cette information en conséquence.

5.2.4.9 Mémentos de fronts de signaux

Position : 4e mot de données du bloc d'états de signaux (voir Tableau 5).

Les mémentos de fronts servent à détecter les changements d'événement survenus (changements d'alarme). Ils ne sont pas configurés, mais ils sont évalués dans le système d'alarmes S5.

5.2.4.10 Structure du bloc de données de paramètres

Pour les types d'alarmes 2 à 4, il faut, outre un bloc de données d'offset, encore configurer des blocs de données de paramètres contenant les données complémentaires d'une l'alarme. L'état de signal d'une alarme est rangé dans un bloc de données d'offset. Les adresses des blocs de données de paramètres sont stockées dans des blocs de données dont la numérotation continue commence immédiatement à la suite du bloc de données d'offset correspondant.

Relation entre bloc de données d'offset et bloc de données de paramètres :

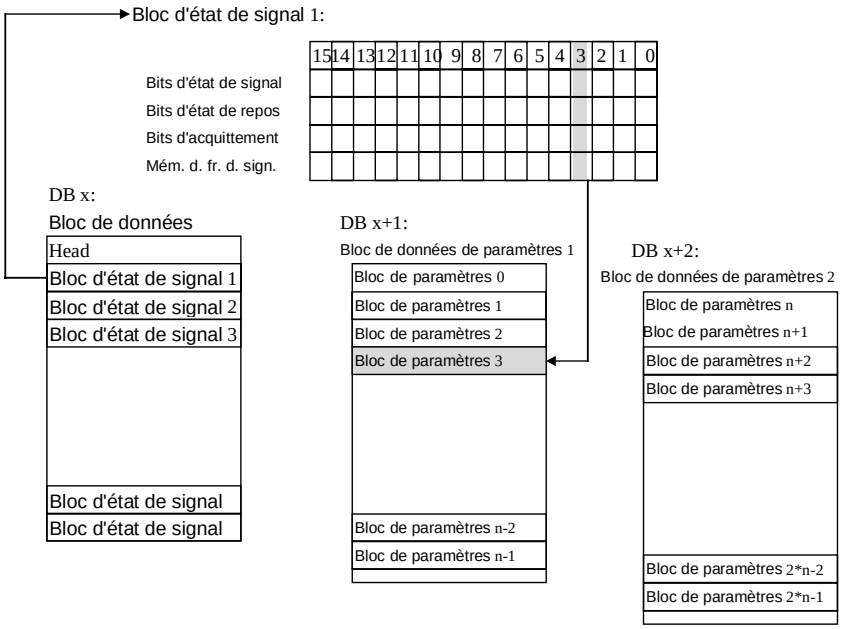


Figure 3

Type	Nombre maxi	Taille du bloc de paramètres	Nombre de blocs de PM par DB de paramètres	Nombre maxi de DB de paramètres
1	1008 / 2016	-	-	-
2	1008	2 DW	128	8
3	1008	5 DW	51	20
4	1008	7 DW	36	28

Tableau 8

Calcul du nombre de blocs de données de paramètres :

$$\text{Anzahl Parameter Datenbaustein} = \frac{\text{benutzte Meldungen}}{\text{Anzahl Parameterblöcke je Parameter Datenbaustein}}$$

S'assurer lors de la configuration que les adresses ne sont pas déjà utilisées pour un autre type d'alarme et que le nombre des blocs de données de paramètres permettra d'éventuelles extensions futures.
Un bloc de données de paramètres contient des blocs de paramètres affectés aux différentes alarmes. Les blocs de paramètres sont rangés en continu dans le bloc de données de paramètres à partir du bloc de paramètres correspondant à la première alarme de ce type d'alarmes. Les blocs de paramètres sont encore numérotés en continu au-delà des limites d'un DB de paramètres. Lorsque la fin d'un DB de paramètres est atteinte, le bloc de paramètres suivant est rangé au numéro suivant, à partir du DW 0 du DB de paramètres suivant. Le système ne stocke toujours que des blocs de paramètres entiers dans un bloc de données de paramètres.

Calcul de l'adresse de début d'un bloc de paramètres :

Numéro d'offset = *numéro d'alarme - numéro d'alarme de base*
DB paramètres = *DB d'offset + 1 + (numéro d'offset / blocs de paramètres par DB de paramètres)*
Adresse de début DB de paramètres = *(numéro d'offset % blocs de paramètres par DB de paramètres) * Taille de blocs de paramètres*

L'utilisateur doit fournir à l'adresse correspondante les données (variables de process, numéro d'ordre de fabrication, désignation de charge).

5.2.4.11 Structure d'un bloc d'alarme

Un bloc d'alarme envoyé au système de niveau supérieur **WinCC** est constitué de la succession de plusieurs mots de données contenant toutes les informations spécifiques aux alarmes. L'ensemble des mots de données forme un bloc de données. La taille des blocs de données est variable selon les différents types d'alarmes (voir Tableau 10).

Un bloc d'alarme est toujours constitué de deux mots de données au moins indépendamment du type d'alarmes. Il s'agit du numéro d'alarme et de l'état d'alarme. La taille maximale d'un bloc d'alarme est de 12 mots de données, que les alarmes comportent l'horodatage (3 mots de données) et les paramètres correspondants ou non.

DW	Description :
1er DW	Numéro d'alarme
2e DW	Etat d'alarme
3e DW	Heure
4e DW	Heure
5e DW	Date
6e DW	Variable de process

DW	Description :
7e DW	Variable de process
8e DW	Numéro de commande de fabrication
9e DW	Numéro de commande de fabrication
10e DW	Désignation de charge
11e DW	Réserve
12e DW	Réserve

Tableau 9

Lorsque les alarmes ne comportent pas d'horodatage, les mots de données correspondants, normalement prévus aux positions 3 à 5, sont absents. Les mots de données de paramètres sont alors écrits directement à la suite du mot de données d'état. La taille d'un bloc d'alarme (nombre de DW) varie selon le type d'alarme et l'horodatage souhaité ; elle est indiquée dans Tableau 10.

Détermination de la longueur des blocs d'alarme en fonction du type d'alarmes:

Type	Longueur du bloc d'alarme en DW sans horodatage	Longueur du bloc d'alarme en DW avec horodatage
1	2	5
2	4	7
3	7	10
4	9	12

Tableau 10

5.2.4.12 Numéro d'alarme

A chaque alarme est affecté un numéro d'alarme permettant de l'identifier sans équivoque.

5.2.4.13 Etat d'alarme

L'état d'alarme est formé comme suit :

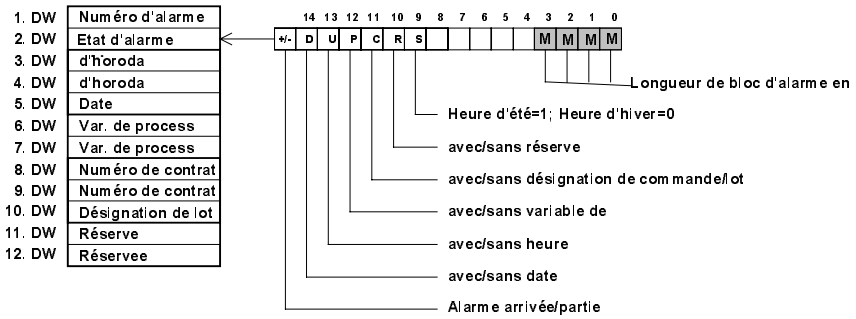


Tableau 11

5.2.4.14 Horodatage

La date et l'heure sont fournies en code binaire par le bloc fonctionnel **FB 86 : MELD:UHR** : Voir chapitre 5.2.10 Fonctions et propriétés de la synchronisation de l'heure S5.

5.2.4.15 Variables de process

Deux mots de données servant à acquérir les valeurs de process à l'arrivée d'une alarme et à les transmettre au système process.

5.2.4.16 Numéro d'ordre de fabrication / désignation de charge

Les deux premiers mots de données s'interprètent, en fonction de la configuration, soit comme code binaire à 32 bits signé soit comme un groupe de quatre caractères ASCII. Le troisième mot de données est à interpréter comme deux caractères ASCII.
Ces trois mots de données permettent de transmettre au système **WinCC** le numéro de commande de fabrication ou la désignation de charge à l'arrivée d'une alarme.

5.2.4.17 Réserve

Les deux mots de données de réserve du type d'alarmes 4 sont prévus pour d'éventuelles extensions ultérieures mais ne sont pas à l'heure actuelle encore implémentés dans le système **WinCC**.

5.2.4.18 Formation d'un bloc d'alarme

Lorsqu'une alarme a été détectée, le numéro de cette alarme est déterminé, à l'aide de la position de bit courante testée, et rangé comme premier mot de données du bloc d'alarme dans le tampon cyclique. Le masque d'états correspondant aux alarmes entrantes ou sortantes, au type d'alarmes et au choix fait pour l'horodatage est sélectionné et rangé comme deuxième mot du bloc d'alarme dans le tampon cyclique. Si le bit spécifique a été paramétré pour qu'il y ait un horodatage dans le bloc de données système, les deux mots de données ci-dessus sont alors suivis des trois mots de données mis à disposition dans le bloc de données système 80, à partir de l'adresse DW 190, au format PC demandé. Le bloc de paramètres correspondant au type de l'alarme est éventuellement lu dans l'archive de données correspondante (bloc de données de paramètres) et placé dans le tampon cyclique contenant la dernière introduction pour terminer le bloc d'alarme.
Le bit d'état suivant de l'alarme suivante est alors testé. Ceci est effectué jusqu'à ce que toutes les alarmes paramétrées aient été traitées.

5.2.4.19 Tampon cyclique interne (FIFO)

Un tampon cyclique (FIFO) est une mémoire à la fin de laquelle le stockage recommence au début : la zone mémoire forme un anneau. Ceci permet, d'une part, de limiter la taille de la mémoire et, d'autre part, d'obtenir un traitement infini.
Dans le système d'acquisition d'alarmes, ceci a pour conséquence que, lorsque la fin virtuelle du tampon est atteinte avant que les données plus anciennes aient été sorties du tampon plein, les données les plus anciennes sont écrasées par les plus récentes ; des informations sont donc perdues.

Le tampon cyclique définit dans la RAM sert, comme le nom l'indique, de tampon pour les alarmes acquises avant que celles-ci soient transmises au PC. Dans la RAM, le tampon cyclique est un espace mémoire d'au moins deux blocs de données et peut, en fonction du paramétrage, être défini avec une largeur quelconque, limitée uniquement par la taille maximale admissible pour les blocs de données d'un automate programmable ou par les DB encore disponibles du programme utilisateur. L'utilisateur transmet au système d'alarmes un nombre de blocs de données dont il disposera pour l'archivage. Lorsque plusieurs blocs de données sont nécessaires, il est indispensable d'utiliser des blocs de données numérotés en continu. Ainsi, l'utilisateur fournit comme paramètres dans le DB système le numéro DB de début et le numéro DB de fin du tampon. Tous les blocs de données se trouvant entre le DB de début et le DB de fin (y compris les deux blocs de données) constituent la mémoire utilisateur du tampon.

5.2.4.20 Interface de transfert de données au système WinCC

Toutes les alarmes correspondant à un cycle sont d'abord écrites dans le tampon cyclique interne par le système d'alarmes S5.
Les entrées d'alarmes (contenu maximum = un bloc de données) sont, après acquisition des alarmes, transmises à l'interface de transfert si celle-ci est prête à recevoir des données. L'interface de transfert, réalisée sous forme de bloc de données, sert de source de données pour les blocs fonctionnels de transmission (blocs de dialogue CP STEP 5). Les blocs de dialogue CP forment une interface vers le processeur de communication correspondant sur le bus de process utilisé (p. ex. bus SINEC-H1).

Structure de l'interface de transfert :

DW	Contenu
DW 0	Longueur du bloc de données
DW 1	KY = [no AG] , [no CPU]
DW 2	KY = [0] , [nombre des alarmes]
DW 3	Début des données utiles (blocs d'alarme)

Tableau 12

DW 0 :

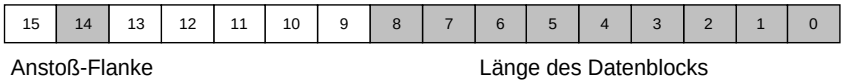


Tableau 13

Le DW 0 de l'interface de transfert fournit tout d'abord dans son bit n° 14 le front de déclenchement d'un contrat souhaité et, dans ses bits n° 0 à 8, la longueur des données source.

Le bloc de données à transmettre ne devant pas dépasser 256 mots de données, et un octet ne pouvant représenter qu'un nombre jusqu'à 255 au maximum, un test séparé des octets avec les commandes DL ou DR n'est pas possible. Il est pour cela recommandé de transférer le DW 0 dans un mémento auxiliaire. Ceci présente en même temps l'avantage que le bit d'autorisation peut être directement évalué individuellement. Cette opération ne peut pas être utilisée avec des mots de données.

Lorsque la condition est remplie, le bit servant de front pour déclencher une fois un contrat d'émission doit être remis à zéro. Les bits restés alors à 1 correspondent à la longueur des données transmises par la source et peuvent être écrits comme QLAE dans la zone de données de paramétrage indirect.
Une fois réussi un contrat d'écriture WRITE (SINEC-H1) dans le système **WinCC** (FOF= terminé sans erreur) le DW 0 de l'interface de transfert doit être écrasé avec la valeur 0. Ceci libère à nouveau l'interface, et d'autres blocs d'alarmes peuvent, s'ils sont présents, être envoyés à l'interface de transfert par le tampon cyclique interne.

Le contrat WRITE (SINEC-H1) doit être réalisé avec la fonction directe SEND et peut être consulté dans le manuel correspondant de l'automate programmable.

5.2.5 Description d'interfaces

Nous allons ci-après décrire les interfaces suivantes :

- Bloc de données système DB 80 :
  pour paramétrage du système d'alarmes S5.
- Bloc de données d'offset pour le type d'alarme correspondant :
  Interface binaire des signaux d'alarmes vers le système d'alarmes S5 avec spécification des propriétés d'alarmes.
- Bloc de données de paramètres pour type d'alarme correspondant :
  pour paramétrage de données d'alarmes complémentaires pour les types d'alarmes 2 à 4.
- Interface de transfert :
  Interface de transfert vers le système **WinCC**.

5.2.5.1 Bloc de données système 80

Le bloc de données système DB 80 permet de configurer des zones de données indépendantes pour 4 types d'alarmes, un tampon cyclique et une interface de transfert. Les mots de données 0 à 20 du DB 80 sont prévus pour la configuration.
Une description détaillée des mots de données 0 à 20 figure au chapitre 5.2.6.

5.2.5.2 Bloc de données d'offset

Le système d'alarmes S5 évalue les états de signaux des alarmes correspondantes et forme à partir de ceux-ci les blocs d'alarmes correspondants.

L'utilisateur doit

- indiquer lors de la configuration les états de repos des différentes alarmes.
- veiller à ce que, pendant l'exécution du programme utilisateur S5, les états d'alarmes soient écrits dans les bits d'états de signaux correspondants.
- veiller à ce que, lorsque c'est nécessaire, les bits d'acquittement soient lus et testés.

5.2.5.3 Bloc de données de paramètres

Pour les types d'alarmes 2 à 4, des informations complémentaires sur l'état courant du process peuvent être transmises par le bloc d'alarme.

L'utilisateur doit

- les variables de process valides (valeur de process, numéro d'ordre de fabrication et numéro de charge) figurent dans les blocs de paramètres correspondants à l'arrivée d'une alarme.

5.2.5.4 Interface de transfert

Le bloc de donnés d'interface de transfert est transmis, dès qu'il contient des blocs d'alarme, directement au système **WinCC** avec un contrat WRITE (SINEC-H1) (voir chapitre 5.2.4.20).

L'utilisateur doit

- les blocs de dialogue CP correspondants de la CPU concernée sont bien disponibles.
- lors de la configuration du système **WinCC**, des canaux de communication correspondants pour une liaison par le bus de process soient indiqués.
- un contrat WRITE soit déclenché comme indiqué au chapitre 5.2.4.20.

5.2.6 Paramétrage du système d'alarmes S5 / DB Système 80

Description des mots de données configurables du bloc de données système DB 80 :

DW	Description
0	Adresse DB : Début FIFO interne
1	Adresse DB : Fin FIFO interne
2	0 : sans horodatage 1 : avec horodatage
3	DB d'offset pour alarmes du type 1
4	1 : un DB d'offset du type 1 2 : deux DB d'offset du type 1
5	DB d'offset pour alarmes du type 2
6	DB d'offset pour alarmes du type 3
7	DB d'offset pour alarmes du type 4
8	Réserve
9	Réserve
10	Adresse DB : Interface de transfert CPU -> PC
11	1 : optimisé pour acquisition (EFOP)
12	EFOP à partir de n alarmes
13	Type AG (115 / 135 / 155)
14	Réserve (doit être 1)
15	N° AG : 1..255; no CPU : 1..4
16	Réserve
17	Réserve
18	Réserve
19	Réserve
20	Mot PAFE du contrôle de vraisemblance

Tableau 14

DW 0, DW 1 : DB Zone de mémoire du tampon cyclique interne
Ces deux mots de données définissent la zone du tampon cyclique interne des alarmes.
L'espace mémoire doit être au moins de deux blocs de données ; veillez à paramétrer la fin du FIFO plus grande que le début du FIFO.
La zone mémoire du tampon est la zone de blocs de données limitée par le début du FIFO et la fin du FIFO, y compris les deux blocs de données indiqués.

Sélection de la taille du tampon cyclique :
Lorsque la capacité mémoire du tampon cyclique est atteinte, les alarmes les plus anciennes sont écrasées. Choisir le nombre de DB assez grand pour que des alarmes ne soient pas écrasées avant leur sortie du tampon en cas d'avalanche d'alarmes. Appliquez la formule approximative suivante pour éviter un tel écrasement.

Détermination du nombre de DB du tampon cyclique :
Alarmes par DB = (255 DW / BD) / longueur de blocs d'alarmes
voir Tableau 10

$$\text{Anzahl der DB} = \frac{\text{Anzahl der Meldungen bei auftretendem Meldungsschwall}}{\text{Meldungen pro DB}}$$

En *acquisition optimisée* il est recommandé de prévoir un à deux blocs de données supplémentaires.

DW 2 : Code d'horodatage
Le choix d'horodater les alarmes concerne toutes les alarmes paramétrées. Soit que toutes les alarmes acquises reçoivent un horodatage (Time stamp) (DW2 = 1) soit qu'aucune alarme n'est horodatée (DW2 = 0). Lorsque DW2 = 0, le système **WinCC** ajoute un horodatage aux blocs d'alarmes qui arrivent.

DW 3, DW 4 : DB d'offset du type d'alarme 1
Lorsque des alarmes du type 1 (alarmes sans paramètres et désignation de charge) doivent être configurées, l'adresse du bloc de données d'offset doit être indiquée dans le mot de données 3. Les états des signaux de ces alarmes doivent être écrits en continu par le programme utilisateur de l'automate dans les blocs de données indiqués.

Si plus de 1008 alarmes (maximum 2016 alarmes) du type 1 sont prévues, entrez le chiffre '2' dans le mot de données 4 pour libérer un autre bloc de données pour les alarmes du type 1. Le deuxième bloc de données a automatiquement l'adresse immédiatement supérieure à celle contenue dans le DW3. Pour un nombre maxi d'alarmes de type 1 = 1008, le chiffre '1' est inscrit dans le DW 4 .

DW 5, DW 6, DW 7 : DB d'offset des types d'alarme 2, 3, 4
Les mots de données 5 à 7 contiennent, de manière analogue au DW 3, les adresses de blocs de données dans lesquels sont rangés les signaux des alarmes.

Le DW 5 contient l'adresse du bloc de données pour le type d'alarme 2, le DW 6 l'adresse du bloc de données pour le type d'alarme 3 et le DW 7 pour le type d'alarmes 4.

Lorsqu'un type d'alarmes n'est pas utilisé, la valeur '0' doit être inscrite dans le DW correspondant.

Les adresses indiquées dans les DW 5-7 sont des 'DB d'offset'. Ceux-ci sont suivis d'un certain nombre de 'DB' qui sont fonction du type d'alarme et du nombre d'alarmes par type. Ces derniers contiennent les paramètres des alarmes. C'est pourquoi il faut faire attention, lors de l'attribution des adresses de DB d'offset qu'il reste suffisamment de place entre les DB d'offset et les DB à indiquer (blocs de données) pour les DB de paramètres.

1008 alarmes peuvent être configurées au maximum pour chacun des types d'alarmes 2 à 4. Si toutes les possibilités sont utilisées, cela donne par conséquent un nombre variable de DB (de paramètres) ajoutés au DB d'offset pour les différents types d'alarmes (voir Tableau 8).

DW 10 : Interface vers le système WinCC
Le mot de données 10 doit toujours être paramétré, quel que soit le mode dans lequel le système d'acquisition des alarmes doit fonctionner. Le DW 10 sert à l'attribution de l'adresse de DB de transfert. Le DB de transfert sert d'interface entre l'automate SIMATIC S5 et le système **WinCC**.

DW 11, DW 12 : Sélection du mode de fonctionnement pour acquisition optimale des alarmes et nombre correspondant d'alarmes
Deux modes de fonctionnement sont prévus :

- '0' dans DW 11 -> **'Fonctionnement normal'** du système d'acquisition des alarmes
- '1' dans DW 11 -> **Mode 'Acquisition optimisée'** du système d'acquisition d'alarmes

Fonctionnement normal :

Un nombre d'alarmes acquises pendant un cycle égal au nombre d'alarmes supportées par l'interface de transfert sont sorties du tampon interne pour émission à condition que l'interface de transfert soit prête à recevoir des données.
Un grand nombre d'alarmes survenant pendant un ou plusieurs cycles successifs entraîneraient un temps de cycle relativement élevé. Celui-ci serait d'autant plus élevé que la taille des blocs d'alarme des types d'alarme concernés est importante. Dans ce cas, l'acquisition des blocs d'alarme serait plus complexe et plus longue.

Acquisition optimisée :

Pour les alarmes qui surviennent, l'acquisition chronologique des alarmes a priorité sur leur transfert au PC. Le temps relatif s'écoulant entre les alarmes émises par le process est prioritaire. Le fait que les alarmes arrivent au PC quelques millisecondes plus tard est secondaire. Le temps d'accomodation et la capacité de perception de l'observateur en salle de contrôle sont les éléments décisifs.

Afin de pouvoir abaisser le temps de cycle du système d'acquisition des alarmes dans les applications à temps critique, le mode 'acquisition optimisée' a été introduit. Le nombre minimum des alarmes devant survenir dans un cycle du bloc OB1 doit être inscrit dans le DW 12. Si le nombre d'alarmes dépasse ce nombre minimum pendant l'exécution du cycle OB1 courant, les alarmes ne sont que saisies et stockées en tampon. Le cycle OB1 renonce alors à la sortie et au transfert à un partenaire connecté dans le cycle OB1.

DW 15 : Numéro AG / Numéro CPU
Ce mot de données est nécessaire pour la constitution de l'en-tête de télégramme et nécessite l'entrée du numéro d'automate programmable (AG) et du numéro de CPU de cet automate pour le projet. Le numéro de la CPU est notamment important lorsque plusieurs CPU fonctionnent sur un automate. Ce n'est qu'en liaison avec ce mot de données, qui contient le code d'alarme, que le système WinCC peut interpréter comme alarme les données transmises, affecter les textes d'alarmes spécifiques aux alarmes et les évaluer.

Le DW 15 est le seul mot de données ayant le format de données 'KY' au paramétrage. Ceci permet de représenter deux octets séparés par une virgule. L'octet de gauche contient le numéro de l'automate, devant être compris dans l'intervalle 1 à 255. Dans l'octet de droite, le numéro de CPU compris entre 1 et 4, est indiqué.

Exemple :

KY = 10,2
Numéro AG = 10
Numéro CPU = 2

DW 20 : Erreur de paramétrage
Tous les mots de données paramétrés dans le DB système subissent un contrôle de vraisemblance au lancement du système d'alarmes S5. Le système distingue ici entre les dépassements de plages de valeurs éventuelles, les affectations multiples de blocs de données paramétrés et les indications manquantes.

Ce bloc de fonction, servant de paramètre de sortie au format d'un mot de données, possède un mot PAFE (mot d'erreur de paramétrage) ; ce dernier est semblable aux blocs de dialogue CP spécifiques au système. L'état du mot PAFE peut être lu dans le DB système 80 (dans le DW 20). Après restitution du contrôle par le FB 81, le mot PAFE est analysé pour voir s'il contient des erreurs. Les actions correctrices peuvent alors être entreprises.

Il est recommandé de faire passer l'automate programmable à l'état 'Stop' en présence d'un mot PAFE différent de zéro. Si le mot PAFE n'est pas pris en compte, la bonne exécution du programme ne peut être garantie.

Evaluation du mot PAFE
Si le programme ou l'automate passe à l'état 'Stop' en présence d'une erreur (mot PAFE <> 0), le numéro d'erreur permet l'analyse ciblée et l'élimination de l'erreur. Le tableau ci-dessous renseigne sur le type d'erreur provoqué au paramétrage.

Format du mot PAFE :

KY = Numéro d'erreur, code d'erreurs groupées

Exemple :

KY = 9,1

Signification de l'erreur de paramétrage numéro 9 :

Adresse DB d'offset du type d'alarme 1 plus grand que adresse DB maxi admissible.

N° erreur	Signification
1	DB de début du tampon interne pas défini
2	DB de début du tampon interne même adresse que le DB système ('80')
3	Adresse DB début du tampon interne plus grande que adresse DB maxi admissible
4	DB fin du tampon interne même adresse que DB système ('80')
5	Adresse DB fin plus petite que adresse DB début du tampon interne
6	Adresse DB fin du tampon interne plus grande que adresse DB maxi
7	DB d'offset type 1 même adresse que DB système ('80')
8	Adresse DB d'offset type 1 dans zone de tampon interne
9	Adresse DB d'offset du type d'alarme 1 plus grand que adresse DB maxi admissible.
10	DB d'offset type 2 même adresse que DB système ('80')
11	DB d'offset type 2 même adresse que type 1
12	DB d'offset type 2 même adresse que 2e DB d'offset type 1
13	Adresse DB d'offset type 2 dans zone de tampon interne
14	Adresse DB d'offset du type d'alarme 2 plus grand que adresse DB maxi admissible.
15	DB d'offset type 3 même adresse que DB système ('80')
16	DB d'offset type 3 même adresse que type 1
17	DB d'offset type 3 même adresse que 2e DB d'offset type 1
18	DB d'offset type 3 même adresse que type 2
19	Adresse DB d'offset type 3 dans zone de tampon interne
20	Adresse DB d'offset du type 3 plus grande que adresse DB maxi admissible.
21	DB d'offset type 4 même adresse que DB système ('80')
22	DB d'offset type 4 même adresse que type 1
23	DB d'offset type 4 même adresse que 2e DB d'offset type 1
24	Adresse DB d'offset type 4 dans zone de tampon interne
25	Adresse DB d'offset du type d'alarme 4 plus grand que adresse DB maxi admissible.
26	DB d'offset type 4 même adresse que type 2
27	DB d'offset type 4 même adresse que type 3
28	Interface de transfert vers PC même adresse que DB système ('80')
29	Interface de transfert vers PC pas définie ('0')
30	Adresse interface de transfert vers PC dans zone de tampon interne
31	Adresse interface de transfert vers PC plus grande que adresse DB maxi

N° erreur	Signification
	admissible.
32	Adresse interface de transfert vers PC même adresse que DB d'offset type 1
33	Adresse interface de transfert vers PC même adresse que DB d'offset type 2
34	Adresse interface de transfert vers PC même adresse que DB d'offset type 3
35	Adresse interface de transfert vers PC même adresse que DB d'offset type 4
36	Adresse interface de transfert vers PC même adresse que 2e DB d'offset type 1
37	DW réserve 9 ou DW réserve 10 différent 0
38	DW réserve 9 ou DW réserve 10 différent 0
39	DW réserve 9 ou DW réserve 10 différent 0
40	DW réserve 9 ou DW réserve 10 différent 0
41	DW réserve 9 ou DW réserve 10 différent 0
42	DW réserve 9 ou DW réserve 10 différent 0
43	DW réserve 9 ou DW réserve 10 différent 0
44	DW réserve 9 ou DW réserve 10 différent 0
45	DW réserve 9 ou DW réserve 10 différent 0
46	DW réserve 9 ou DW réserve 10 différent 0
47	Le nombre d'alarmes mini pour le mode sélectionné avec acquisition optimisée manque
48	Type AG pas défini
49	DW réserve 14 différent 1
50	Numéro AG pour en-tête télégramme pas défini
51	Numéro CPU pour en-tête télégramme pas défini
52	Numéro CPU plus grand que valeur admise (1..4)

Tableau 15

5.2.7 Exemples de configuration du système d'alarmes S5

Description

Le système d'alarmes S5 doit être configuré pour les types d'alarmes suivants :

Type	Définition : Types d'alarmes
1	1200 alarmes (du numéro d'alarme 10000 à 11199) les alarmes 11000 à 11199 sont actives à l'état 'low'
2	Aucune alarme prévue
3	11 alarmes (du numéro d'alarme 30000 à 30010)
4	Aucune alarme prévue

Toutes les alarmes sont pourvues d'un horodatage.
L'automate utilisé est un 135U, numéro AG 1, numéro CPU 1.

5.2.7.1 Paramétrages DB 80

Type	Nombre maxi	Taille du bloc de paramètres	Nombre de blocs de PM par DB de paramètres	Nombre maxi de DB de paramètres
1	1008 / 2016	-	-	-
2	1008	2 DW	128	8
3	1008	5 DW	51	20
4	1008	7 DW	36	28

Le DB 81 est utilisé comme interface de transfert vers le PC.

Pour une apparition régulière des alarmes disponibles, la longueur de bloc d'alarme moyenne (avec horodatage) est de :
*(1200 * 5 + 11 * 10) / (1200 + 11) = 5,04*

Hypothèse :
Le système d'alarmes S5 doit pouvoir supporter des avalanches de 100 alarmes par cycle d'API et traiter au moins 30 alarmes en mode 'Acquisition optimisée'.

*5 DW/Mld. * 100 Mld = 500 DW*
(500 DW) / (256 DW/DB) = 1,95 DB

Cela donne quatre blocs de données pour le tampon cyclique, un à deux blocs de données supplémentaires devant être prévus pour le 'mode acquisition optimisée'.
Le tampon cyclique commence à l'adresse de bloc de données 82, ce qui donne comme adresse de fin de tampon cyclique DB 85.

Afin de prévoir une réserve pour une extension future du tampon cyclique, le bloc de données d'offset d'alarme de type 1 est aux adresses DB 88 et DB 89 (si plus de 1088 alarmes de type 1).

Le DB 90 devient le bloc de données d'offset du type d'alarme 3. Un DB de paramètres du type d'alarme 3 a une capacité de 51 blocs de paramètres ; si l'on soustrait les 11 blocs utilisés, il en résulte qu'un bloc de paramètres (DB 91) permet une extension de 40 alarmes du type 3.

DW	Description	Valeur
0	Adresse DB : Début FIFO interne	82
1	Adresse DB : Fin FIFO interne	85
2	0 : sans horodatage 1 : avec horodatage	1
3	DB d'offset pour alarmes du type 1	88
4	1 : un DB d'offset du type 1 2 : deux DB d'offset du type 1	2
5	DB d'offset pour alarmes du type 2	0
6	DB d'offset pour alarmes du type 3	90
7	DB d'offset pour alarmes du type 4	0
8	Réserve	0
9	Réserve	0
10	Adresse DB : Interface de transfert CPU -> PC	81
11	1 : optimisé pour acquisition (EFOP)	1
12	EFOP à partir de n alarmes	30
13	Type AG (115 / 135 / 155)	135
14	Réserve	1
15	N° AG : 1..255; no CPU : 1..4	1, 1
16	Réserve	0
17	Réserve	0
18	Réserve	0
19	Réserve	0
20	Mot PAFE du contrôle de vraisemblance	0

Le bloc de données 100 est utilisé par les DW 10 à DW 20 pour la synchronisation de l'heure.
Le bloc de données 101 est utilisé par les DW 0 à DW 255 pour la programmation de commandes.

5.2.7.2 Création des blocs de données

Création des blocs de données DB 81 à DB 85, DB 88 à DB 91 et DB 101 de DW 0 à DW 255.
Création du bloc de données DB 100 de DW 0 à DW 20.

5.2.7.3 Initialisation des blocs de données d'offset

Type d'alarme 1
Les DB 88 et 89 sont prévus pour le type d'alarme 1. LE DB 88 contient les alarmes numéros 10000 à 11007, le DB 89 les alarmes 11008 à 11199.
Au total, 1200 alarmes du type 1 doivent être configurées.

Voir chapitre 5.2 Adresse du dernier bloc d'états de signaux :
Numéro d'alarme d'offset = Numéro d'alarme - Numéro d'alarme de base = 0 bis 1199

1er DB d'offset :

Adresse du dernier bloc d'états de signaux DW 252

2e DB d'offset :

Adresse du dernier bloc d'états de signaux DW 252

1200 - 1008 = 192

192 / 16 = 12
192 % 16 = 0

Adresse du dernier
*bloc d'états de signaux dans le bloc de données = 12 * 4 = 48*
d'offset 2

DB 88 :

DW	Description	Valeur
DW 0	sans affectation	
DW 1	Numéro d'alarme de base	10000
DW 2	Adresse du dernier DW	252
DW 3	sans affectation	

DB 89 :

DW	Description	Valeur
DW 0	sans affectation	
DW 1	Numéro d'alarme de base	11018
DW 2	Adresse du dernier DW	48
DW 3	sans affectation	

Voir chapitre 5.2.4.3 Numéro d'offset / Etats des signaux des alarmes :

Les alarmes 11000 à 11199 sont actives à l'état 'low'.

Position du bit d'état de repos du numéro d'alarme 11000 :

Numéro d'offset	*11000 - 10000 = 1000*
Début du bloc d'états de signaux	*(numéro d'offset / 16 + 1) * 4 =* *= (62 + 1) * 4 * DW 252*
Mot de données des bits d'états de repos :	*DW 253*
Bit de données :	*numéro d'offset % 16 = 8*
Bloc de données	*Bloc de données d'offset = DB 88*

Position du bit d'état de repos du numéro d'alarme 11000 :

Numéro d'offset	*11199 - 10000 = 1199*
Début du bloc d'états de signaux	*(numéro d'offset / 16+ 1) * 4 =*
	*= (74 +1) * 4 = 300*
	300 - 252 = 48
Mot de données des bits d'états de repos :	*DW 49*
Bit de données :	*numéro d'offset % 16 = 15*
Bloc de données	*Bloc de données d'offset + 1 = DB 89*

Les bits d'état de repos ci-dessous doivent être modifiés :

DB 88 :

DW 253 : bits de données 8 à 15 mis à '1'

DB 89 :

DW 5, DW 9, DW 13, ... DW 49 : bits de données 0 à 15 mis à '1'

Type d'alarme 3
Pour le type d'alarme 3, le DB 90 est prévu comme bloc de données d'offset pour les alarmes 30000 à 30010 et le DB 91 pour le bloc de données de paramètres.
Au total, 11 alarmes du type 3 doivent être configurées.
Voir chapitre 5.2.4.10 Structure du bloc de données de paramètres:

Numéro d'offset = Numéro d'alarme - Numéro d'alarme de base = 0 à 10

DB d'offset :

Adresse du dernier bloc d'états de signaux		*11 / 16 = 0*
		11 % 16 = 11
Adresse du dernier bloc d'états de signaux	=	*(0+1) * 4 = 4*

DB 89 :

DW	Description	Valeur
DW 0	sans affectation	
DW 1	Numéro d'alarme de base	30000
DW 2	Adresse du dernier DW	4
DW 3	sans affectation	

Tous les bits d'état de repos sont à 0
Voir chapitre 5.2.4.10 Structure du bloc de données de paramètres

DB paramètres = DB d'offset + 1 + (numéro d'offset / blocs de paramètres par DB de paramètres)

Numéro d'alarme 30000 :
DB paramètres = 90 + 1 + 0 / 51 = 91

Numéro d'alarme 30010 :
DB paramètres = 90 + 1 + 10 / 51 = 91

Adresse de début du bloc de paramètres correspondant : = *(numéro d'offset % blocs de paramètres par DB de paramètres) * Taille de blocs de paramètres*

Numéro d'alarme 30000 :
Adresse de début du bloc de paramètres correspondant : = *(0 % 51) * 5 = DW*

Numéro d'alarme 30010 :
Adresse de début du bloc de paramètres correspondant : = *(10 % 51) * 5 = DW 50*

DB 91 : Bloc de données de paramètres 91 du bloc de données d'offset 90

N° d'alarme	Valeurs de process	Numéro d'ordre de fabrication	Désignation de charge
30000	DW 0, 1	DW 2, 3	DW 4
30001	DW 5, 6	DW 7, 8	DW 9
30002	DW 10, 11	DW 12, 13	DW 14
30003	DW 15, 16	DW 17, 18	DW 19
30004	DW 20, 21	DW 22, 23	DW 24
30005	DW 25, 26	DW 27, 28	DW 29
30006	DW 30, 31	DW 32, 33	DW 34
30007	DW 35, 36	DW 37, 38	DW 39
30008	DW 40, 41	DW 42, 43	DW 44
30009	DW 45, 46	DW 47, 48	DW 49
30010	DW 50, 51	DW 52, 53	DW 54

5.2.8 Documentation des blocs de commande SIMATIC S5

Tâche et mode de fonctionnement des blocs de commandes S5

Le logiciel sert à 'traiter' les bits, octets, mots et doubles mots dans les automates SIMATIC S5 via un bus de process (p. ex. SINEC H1). Le bus permet uniquement l'adressage d'octets et de mots dans l'automate SIMATIC S5.
Les opérations suivantes sont exécutables en standard :

- Les blocs de données (DB et DX), temporisateurs et compteurs ne peuvent être modifiés qu'au niveau mot.
- Les mémentos, entrées, sorties, périphéries (P et Q) ne peuvent être modifiés qu'au niveau de l'octet.

Ce package fournit, dans SIMATIC S5, les fonctionnalités logicielles nécessaires à la réalisation des opérations ci-dessous par le système WinCC à travers le bus de process.

- Activation d'une impulsion de sens pour le temps d'exécution d'un cycle OB1
- Mise à 1 / mise à 0 / inversion de bit dans les blocs DB / DX
- Mise à 1 / mise à 0 / inversion de bit dans mémentos
- Ecriture octet gauche / droit dans les blocs DB / DX
- Ecriture mot / double mot dans les blocs DB / DX
- Ecriture octet / mot dans mémentos
- Ecriture octet / mot dans périphérie
- Ecriture octet / mot dans périphérie étendue

Les modifications souhaitées dans l'automate SIMATIC S5 sont mises à disposition comme variables de données brutes par une interface de données de WinCC - Control Center. Les commandes doivent être envoyées à l'automate S5 par la variable de données brutes. Ces ordres sont évalués et exécutés directement dans l'automate S5 par un bloc interpréteur de commandes **FB 87 : EXECUTE**.

Le présent manuel décrit de manière détaillée l'utilisation et la manipulation des blocs de commandes S5 dans l'environnement SIMATIC S5. Une vue d'ensemble des fonctions et des blocs de données utilisés par le logiciel ainsi que des espaces mémoire nécessaires est présentée à l'utilisateur. Vient ensuite une description détaillée de l'interface de données disponible. Un exemple de configuration est présenté pour faciliter la prise en main.

5.2.8.1 Liste des blocs logiciels

Le logiciel SIMATIC S5 *'S5 - blocs de commande'* est livré sur le CD-ROM WinCC, dans le fichier **WINCC1ST.S5D**.
Ce fichier contient les blocs fonctionnels suivants pour les 'blocs de commande S5' :

FB	Nom	Taille en octets	Fonction
FB 87	EXECUTE	152	Permet la manipulation de bits, octets, mots, doubles mots par le bus de process
FB 88	OPCODE	399	Appelé par FB 87
Total		551	

Tableau 16

Un bloc de données de commande de 512 octets est en outre nécessaire.

5.2.8.2 Conditions à remplir par le matériel

Les blocs fonctionnels indiqués dans le Tableau 16 nécessitent pour leur exécution correcte le matériel suivant :

AG	CPU
AG 115U	CPU 943, CPU 944, CPU 945
AG 135U	CPU 928A, CPU 928B
AG 155U	CPU 946/ 947, CPU 948

5.2.8.3 Paramètres d'appel de FB 87 : EXECUTE

Les paramètres d'appel du bloc fonctionnel **FB 87 : EXECUTE** sont décrits ci-dessous.

Nom	EXECUTE	Paramètres
Dés :	DBNR	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KF
Dés :	DMDX	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KF
Dés :	RIMP	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KY

DBNR : Numéro de bloc de données de l'interface de transfert de commandes

DDWX : Type de la source de données pour l'interface de transfert de commandes

DB.....la source de données est un bloc de données (DB).
DX.....la source de données est un bloc de données étendu (DX).

RIMP : Position binaire pour l'impulsion de sens

RIMP.....numéro de mémento, numéro de bit

5.2.9 Description d'interfaces

Nous allons ci-après décrire les interfaces suivantes :

- Bloc fonctionnel de commande FB 87
- Bloc de données de commande : Interface de transmission de commandes vers SIMATIC S5.

Dans l'automate SIMATIC S5, l'interpréteur de commandes (**FB 87 : EXECUTE**) est appelé cycliquement dans le bloc OB 1. Le type et l'adresse du DB de commandes sont passés comme paramètres. En présence d'une commande, le code Op (Opcode) est transmis avec quatre paramètres au **FB 88 : OPCODE** et directement exécuté. Lorsque la commande a été exécutée, le compteur de commandes (DW 1) est décrémenté de la valeur 1. La procédure de transfert de commandes et de décrémentation du compteur de commandes est répétée jusqu'à ce que toutes les commandes présentes aient été exécutées.

Les indications de type et d'adresse du bloc de données peuvent être identiques dans WinCC Control-Center et dans le programme S5 ; le bloc de données doit être dans l'automate S5. Il est possible de choisir entre un bloc de données DB et un bloc DX et son adresse (p. ex. DX 234). Le bloc de données doit être ouvert par l'utilisateur jusqu'au mot de données 255, les mots de données 0-255 pouvant être adressés dans le bloc de données indiqué.

Syntaxe des commandes rangées dans le bloc de données de commande :

DW	Description
0	sans affectation
1	Nombre des commandes à exécuter
2	Opcode de la première commande
3	Paramètre 1 (Opcode 1)
4	Paramètre 2 (Opcode 1)
5	Paramètre 3 (Opcode 1)
6	Paramètre 4 (Opcode 1)
7	Opcode de la deuxième commande
8	Paramètre 1 (Opcode 2)
9	Paramètre 1 (Opcode 2)
10	Paramètre 2 (Opcode 2)
11	Paramètre 3 (Opcode 2)
12	Paramètre 4 (Opcode 2)
13	Opcode de la troisième commande
14	Paramètre 1 (Opcode 3)
.....	

Ci-dessous la description syntaxique des commandes implémentées :

Passage du code Op et des paramètres dans DB de commande

Commande	Code Op	Paramètre 1	Paramètre 2	Paramètre 3	Paramètre 4
Mettre bit à '1' dans DB	10	DB	DW	Bit	-
Mettre bit à 0 dans DB	11	DB	DW	Bit	-
Inverser bit dans DB	12	DB	DW	Bit	-
Mettre à '1' octet de droite dans DB	15	DB	DW	Valeur	-

Commande	Code Op	Paramètre 1	Paramètre 2	Paramètre 3	Paramètre 4
Mettre à '1' octet de gauche dans DB	16	DB	DW	Valeur	-
Ecrire mot de données dans DB	17	DB	DW	Valeur	-
Ecrire double mot de données dans DB	18	DB	DW	Valeur	Valeur
Mettre bit à '1' dans DX	20	DX	DW	Bit	-
Mettre bit à 0 dans DX	21	DX	DW	Bit	-
Inverser bit dans DX	22	DX	DW	Bit	-
Mettre à '1' octet de droite dans DX	25	DX	DW	Valeur	-
Mettre à '1' octet de gauche dans DX	26	DX	DW	Valeur	-
Ecrire mot de données dans DX	27	DX	DW	Valeur	-
Ecrire double mot de données dans DX	28	DX	DW	Valeur	Valeur
Mettre à '1' bit de mémento	30	MB	Bit	-	-
Mettre à 0 bit de mémento	31	MB	Bit	-	-
Inverser bit de mémento	32	MB	Bit	-	-
Ecrire octet de mémento	35	MB	Valeur	-	-
Ecrire mot de mémento	36	MW	Valeur		-
Ecrire octet de périphérie	45	PB	Valeur	-	-
Ecrire mot de périphérie	46	PW	Valeur	-	-
Ecrire octet de périphérie étendue	55	QB	Valeur	-	-
Ecrire mot de périphérie étendue	56	QW	Valeur	-	-
Mettre à '1' impulsion de sens	60	-	-	-	-

5.2.9.1 Exemples de configuration pour les blocs de commandes S5

Il s'agit de créer les blocs de commande S5.
L'impulsion de sens est mise à disposition dans le mot de mémentos 56, bit 4. Le bloc DX 237 sert de bloc de données de commande. S'assurer pour cela que le bloc de données DX 237 est ouvert, du mot de données DW 0 au mot de données DW 255, sur l'automate.
Dans WinCC Control-Center, il est nécessaire d'entrer le bloc de données souhaité lors du paramétrage des paramètres de canaux (p. ex. SINEC H1).

Extrait de OB 1 :

```
......
: SPA  FB 87
NAME   : EXECUTE
DBNR :  KF +237
DDWX :  KC DX
RIMP :  KY 56, 4
......
```

5.2.10 Fonctions et propriétés de la synchronisation de l'heure S5

La présente documentation décrit les fonctions et les propriétés du logiciel SIMATIC S5 :

Synchronisation de l'heure S5

Le logiciel sert à synchroniser l'heure système SIMATIC S5. Il fournit en outre un format d'horodatage adapté à 'l'acquisition chronologique des alarmes' du système d'alarmes S5 pour la constitution des blocs d'alarmes.

Le bloc fonctionnel **FB 86 : MELD:UHR** fournit en outre l'heure S5 courante dans un format nécessaire à l"acquisition chronologique des alarmes'. Les données sont rangées dans le bloc de données système 80, à partir du mot de données DW 190.

En présence d'un changement d'état d'un signal d'alarme, l'alarme correspondante est identifiée à l'aide du numéro d'alarme par le bloc fonctionnel **FB 80 : SYSTEMFB** et reçoit un horodatage (Time stamp) du bloc de données système 80.

Le présent manuel décrit de manière détaillée l'utilisation et la manipulation de la synchronisation de l'heure S5 dans l'environnement SIMATIC S5. Une vue d'ensemble des fonctions et des blocs de données utilisés par le logiciel ainsi que des espaces mémoire nécessaires est présentée à l'utilisateur. Un exemple de configuration est présenté pour faciliter la prise en main.

5.2.10.1 Liste des blocs logiciels

Le logiciel SIMATIC S5 (synchronisation de l'heure S5) se trouve sur le CD-ROM WinCC, dans le fichier : WINCC1ST.S5D.

Ce fichier contient les blocs de données et blocs fonctionnels suivants :

FB	Nom	Taille en octets	Fonction
FB 86	MELD:UHR	1135	Synchronisation de l'heure
Total		1135	

Tableau 17

Zone de données d'heure 115U : 27 DW = 54 octets
Zone de données d'heure 135/155U : 12 DW = 24 octets

Zone de données pour système d'alarmes S5 : 3 DW = 6 octets

5.2.10.2 Conditions à remplir par le matériel

Les blocs fonctionels indiqués pour le système d'alarmes S5 nécessitent, pour l'exécution correcte des fonctions, le matériel suivant :

AG	CPU
AG 115U	CPU 944 * , CPU 945
AG 135U	CPU 928B
AG 155U	CPU 946/ 947, CPU 948

* seule la CPU 944 dotée de deux interfaces PG possède une heure système

5.2.11 Paramètres d'appel de FB 86 : MELD:UHR

Les paramètres d'appel du bloc de fonction **FB 86 : MELD:UHR** sont décrits ci-dessous.

Les paramètres d'appel du bloc fonctionnel **FB 87 : EXECUTE** sont décrits ci-dessous.

Nom	MELD:UHR	Paramètres
Dés :	CPUT	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KF
Dés :	DCF7	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KF
Dés :	QTYP	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KF
Dés :	QSYN	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KY
Dés :	UDAT	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KY
Dés :	ZINT	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KF
Dés :	ZUHR	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KY
Dés :	ZSYN	E/A/D/B/T/Z : D KM/KH/KY/KC/KF/KT/KZ/KG : KF

CPUT :

Numéro CPU	Type
1	CPU 943 / CPU 944
2	CPU 945
3	CPU 928B
4	CPU 946 / 947
5	CPU 948

DCF7 :

Mode de fonctionnement
0 = fonctionnement avec heure système S5
1 = fonctionnement avec heure radio DCF77

QTYP :

Type de source de données pour le télégramme de synchronisation de l'heure.
0= la source de données est un bloc de données (DB).
1 = la source de données est un bloc de données étendu (DX).

QSYN :

Source des données d'heure

DCF7 = 0 :	QSYN =	Numéro DB, numéro DW du télégramme de synchronisation d'heure reçu
DCF7 = 1 :	QSYN =	Numéro DB, numéro DW de l'heure radio DCF77

UDAT :

Adresse de la zone de données d'heure
UDAT = numéro DB, numéro DW

ZINT :

Intervalle en minutes pour l'émission du télégramme de synchronisation (DCF7 = 1)

ZUHR :

Zone de données cible pour les données d'heure au format du système d'alarmes.
ZUHR = numéro DB, numéro DW

ZSYN :

Zone de données cible pour le télégramme de synchronisation de l'heure (DCF7 = 1)

Lorsque la fonctionnalité 'Acquisition chronologique d'alarmes' doit être utilisée avec le système d'alarmes S5, un format de données d'heure spécial est attendu dans le DB 80, à partir du mot DW 190.
Ce format de données d'heure est fourni par l'heure système S5 et écrit dans la zone de données correspondante ZUHR (DB 80, DW 190-192).

Relation entre l''acquisition chronologique d'alarmes' et FB 86 : MELD:UHR :

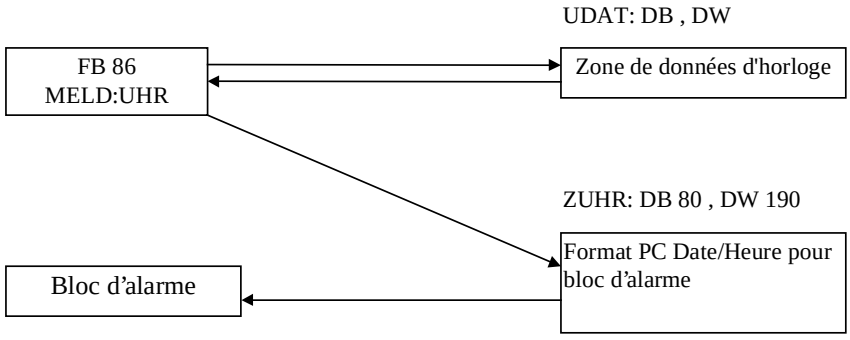


Figure 4

5.2.12 Formats de données pour l'heure et la date

Télégramme de synchronisation de l'heure interne à un système (à l'heure actuelle, WinCC ne supporte pas le télégramme d'heure !!)

Le premier mot de données du télégramme de synchronisation de l'heure contient un code de source de données qui est envoyé avec les données d'horodatage par le système.
Le bloc fonctionnel **FB 86 : MELD:UHR** n'accède au télégramme présent que lorsque le code 'FFFF' est contenu à cet endroit. La réception du télégramme est acquittée par un '0' dans ce mot de données. Il n'est lu et évalué qu'à la réception d'un nouveau télégramme (DW 1 = 'FFFF').

Signification	Mot de données	Contenu	Plage de validité	Remarque
Code de source / télégramme d'heure	1	FFFF		
ID Télégramme	2	FFFF		sans affectation
secondes	3	00xx	xx : 0..59	
minutes	4	00xx	xx : 0..59	
heures	5	00xx	xx : 0..23	
jour	6	00xx	xx : 1..31	
mois	7	00xx	xx : 1..12	
année	8	00xx	xx : 0..127 (1990-2117)	année + 1990
jour de la semaine	9	00xx	xx : 0..6	dimanche = 0
jour de l'année	10	00xx	xx :1..365	
heure d'été, heure d'hiver année bissextile	11	yyxx	xx : heure d'hiver = 00 heure d'été = 01 yy : année bissextile année courante = 00 année précédente = 01 deux ans auparavant = 02 trois ans auparavant = 03	

Tableau 18

5.2.12.1 Zone des données d'heure CPU 944, CPU 945

Les numéros de mots de données sont des numéros relatifs. L'adresse réelle de la zone est déterminée par le paramètre d'appel : UDAT = n° DB, n° DW du bloc fonctionnel **FB 86 : MELD:UHR.**

DW	Zone
0	Variables internes
1	
2	
3	
4	Heure courante
5	
6	
7	
8	Plage de réglage de l'heure
9	
10	
11	
12	Réserve - Alarme d'horloge
13	
14	
15	
16	Réserve - Heures de fonctionnement
17	
18	
19	
20	
21	
22	Heure / Date après RUN / STOP
23	
24	
25	
26	

Tableau 19

Heure courante dans zone de données d'heure :

DW	mot/gauche	mot/droite
4	---	jour de la semaine
6	jour	mois
7	année	AM/PM (bit, n° 7), heure
8	minute	seconde

Figure 5

Plage de réglage dans la zone de données d'heure :

DW	mot/gauche	mot/droite
9	année bissextile	jour de la semaine
10	jour	mois
11	année	AM/PM (bit, n° 7), heure
12	minute	seconde

Figure 6

5.2.12.2 Zone de données d'heure CPU 928B, CPU 948

Les numéros de mots de données sont des numéros relatifs. L'adresse réelle de la zone en mémoire est déterminée par le paramètre d'appel : UDAT = n° DB, n° DW du bloc fonctionnel **FB 86 : MELD:UHR** .

DW	Zone
0	Variables internes
1	
2	
3	
4	Heure courante
5	
6	
7	
8	Plage de réglage de l'heure
9	
10	
11	

Figure 7

Heure courante dans zone de données d'heure :

DW	mot/gauche								mot/droite							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	secondes								0							
5	format		heures						minutes							
6	jour du mois								jour de la semaine				0			
7	année								seconde							

Figure 8

Heure courante dans la plage de réglage :

DW	mot/gauche								mot/droite							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
8	secondes								0							
9	format		heures						minutes							
10	jour du mois								jour de la semaine				0			
11	année								seconde							

Figure 9

5.2.12.3 Zone des données d'heure CPU 946, CPU 947

Les numéros de mots de données sont des numéros relatifs. L'adresse réelle de la zone en mémoire est déterminée par le paramètre d'appel : UDAT = n° DB, n° DW du bloc fonctionnel **FB 86 : MELD:UHR** .

DW	Zone
0	Variables internes
1	
2	
3	
4	Heure courante
5	
6	
7	
8	Plage de réglage de l'heure
9	
10	
11	

Figure 10

Heure courante dans zone de données d'heure :

DW	mot/gauche		mot/droite	
4	dizaine de s	s	1/10 s	1/100 s
6	dizaine d'heures	heure	dizaine de min	min
7	dizaine de jours	jour	jour de la semaine	0
8	dizaine d'années	année	dizaine de mois	mois

Figure 11

Heure courante dans la plage de réglage :

DW	mot/gauche		mot/droite	
9	dizaine de s	s	1/10 s	1/100 s
10	dizaine d'heures	heure	dizaine de min	min
11	dizaine de jours	jour	jour de la semaine	0
12	dizaine d'années	année	dizaine de mois	mois

Figure 12

5.2.12.4 Format de données d'heure pour les blocs d'alarmes

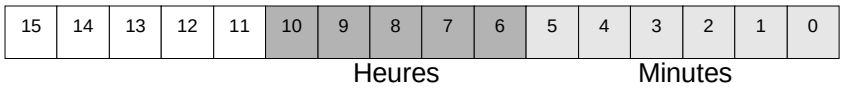
Les numéros de mots de données sont des numéros relatifs. L'adresse réelle de la zone en mémoire est déterminée par les paramètres : ZUHR = n° DB, n° DW du bloc fonctionnel **FB 86 : MELD:UHR** . Lorsque la fonctionnalité 'Acquisition chronologique d'alarmes' doit être utilisée avec le système d'alarmes S5, il est nécessaire d'entrer les données DB 80, DW 190 dans le paramètre ZUHR.
La date et l'heure sont fournies par le bloc fonctionnel **FB 86 : MELD:UHR** pour traitement des alarmes en code binaire :

Heure courante dans la plage de réglage :

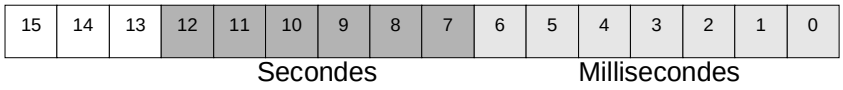
Signification	Mot de données	Bit	Plage de validité	Remarque
1/100 seconde	1	0 - 6	0..99 (0 - 990 ms)	temps de cycle 10 ms
secondes	1	7 - 12	0..59	
minutes	0	0 - 5	0..59	
heures	0	6 - 10	0..23	
jour	2	0 - 4	1..31	
mois	2	5 - 8	1..12	
année	2	9 - 15	0..127 (1990-2117)	année + 1990

Figure 13

DW3 : heure



DW4 : heure



DW4 : date

Figure 14

5.2.13 Description d'interfaces

Pour pouvoir utiliser le logiciel de synchronisation d'heure S5, l'utilisateur doit :

- Renseigner les paramètres d'appel de **FB 86 : MELD:UHR** selon les indications du chapitre 5.2.11 Paramètres d'appel de FB 86 : MELD:UHR
- Ouvrir les zones de données dans l'automate.

Exemple de configuration

Soit une CPU 944 avec deux interfaces PG. La synchronisation d'heure S5 doit être paramétrée sans heure DCF77 sur cette CPU pour le système d'alarmes S5.

Zones de données :

Télégramme de synchronisation de l'heure :	DB 100, DW 20 - DW 30
Zone de données d'heure de l'heure système S5 :	DB 100, DW 31 - DW 47
Données de bloc d'alarmes* :	DB 80, DW 190 -DW 192

* Cette zone de données est prescrite lorsqu'on utilise le système d'alarmes SIMATIC S5.

Lors de la définition des paramètres de canaux (p. ex. SINEC H1), il est nécessaire d'entrer le bloc de données souhaité (DB 100, DW20 - DW 30) dans le système au paramétrage de la 'synchronisation d'heure'.

S'assurer pour cela que le DB 80 est ouvert, du mot de données DW 0 au mot de données DW 255, et le DB 100, du mot de données DW 0 au mot de données DW 47.

Extrait de OB 1 :

```
......
: SPA  FB 86
NAME   : MELD : UHR
CPUT :  KF +1
DCF7 :  KF +0
QTYP :  KF +0
QSYN :  KY 100, 20
UDAT :  KY 100, 31
ZINT :  KF +0
ZUHR :  KY 80, 190
ZSYN :  KF +0
......
```

5.2.14 Interaction avec le système d'alarmes WinCC

Observer les indications ci-dessous concernant l'interaction du système d'alarmes WinCC avec le bloc d'alarmes S5 :

Dans S5, entrez la valeur 256 comme nombre de mots de données à transmettre dans le bloc de transfert.

Dans Control-Center, une nouvelle liaison doit être créée dans canal S5Trsp. Paramétrez le type 'Fetch-passif' pour le point Liaison/Fonction Read.

Deux variables de données brutes doivent être créées par automate pour l'échange de données avec le système d'alarmes.
La première variable de données brutes est destinée à la réception des alarmes.
Paramétrez son adressage comme suit : zone de données : DB, n° DB : xx, mot d'adressage, DW : 0, type de données brutes : événement.
La deuxième variable de données brutes est nécessaire pour l'émission des informations d'acquittement.
Paramétrez son adressage comme suit : zone de données : DB, n° DB : 80, mot d'adressage, DW : 90, type de données brutes : événement.

Dans le système d'alarmes, liez la variable d'événement à la variable de données brutes de réception (l'information de bit est ici sans affectation).
Liez la variable d'acquittement à la variable de données brutes d'émission (l'information de bit est ici sans affectation).
Paramétrez le fichier S5STD.NLL comme DLL de normalisation.
Conseil : l'assistant de connexion permet de lier d'un seul coup toutes les alarmes concernées.

Seuls les nombres positifs à virgule fixe sont admissibles pour les valeurs de process. Les valeurs à virgule flottante ne sont pas supportées.

S5	WinCC
Valeur de process	⇨ Valeur de process 1
Désign. de contrat	⇨ Valeur de process2
Désignation de lot	⇨ Valeur de process 3
Réserve	⇨ Valeur de process 4

5.3 DLL de normalisation d'interface vers AlarmLogging et TagLogging

Finalité

Les applications AlarmLogging et TagLogging acquièrent les données de process par l'intermédiaire du gestionnaire de données de WinCC. En fonction du type de communication avec le process utilisé

- plusieurs DLL de canal participent au transfert des données,
- les données de process contenues dans les télégrammes (variables de données brutes) sont stockées avec une structure variable.

AlarmLogging et TagLogging doivent cependant pouvoir traiter les données de process de la même manière indépendamment du type de communication employé. On utilise donc pour chaque type de communication une DLL de normalisation spécifique qui connaît la structure exacte des divers télégrammes et en dérive un «format de données de process générique» pour AlarmLogging et TagLogging.

Une DLL de normalisation est toujours attachée à une DLL de canal ; comme celle-ci, la DLL de normalisation doit pouvoir être ajoutée au système global et en être enlevée. Elle n'a cependant pas d'interface directe avec la DLL de canal correspondante.

Ce document décrit l'intégration et l'interface de chaque DLL de normalisation aux applications WinCC AlarmLogging et TagLogging. Cette documentation a été élaborée lors de la conception de la DLL de normalisation S7PMC ; c'est pourquoi „DLL de normalisation S7PMC“ doit être dans la plupart des cas compris comme „DLL de normalisation“.

Principe de fonctionnement

La DLL de normalisation S7PMC est une entité de programme passive ne possédant des interfaces que vers les applications AlarmLogging et TagLogging. La DLL de normalisation S7PMC exécute des fonctions S7PMC spéciales pour AlarmLogging et TagLogging.

AlarmLogging et TagLogging se déclarent à la DLL de normalisation par un appel de lancement. Certains paramètres sont alors transmis dans une structure de lancement à la DLL de normalisation Norm.DLL et ses propriétés sont prises en compte à l'aide de codes.

Deux sens de transfert de données sont nécessaires pour l'exécution des fonctions S7PMC en runtime:

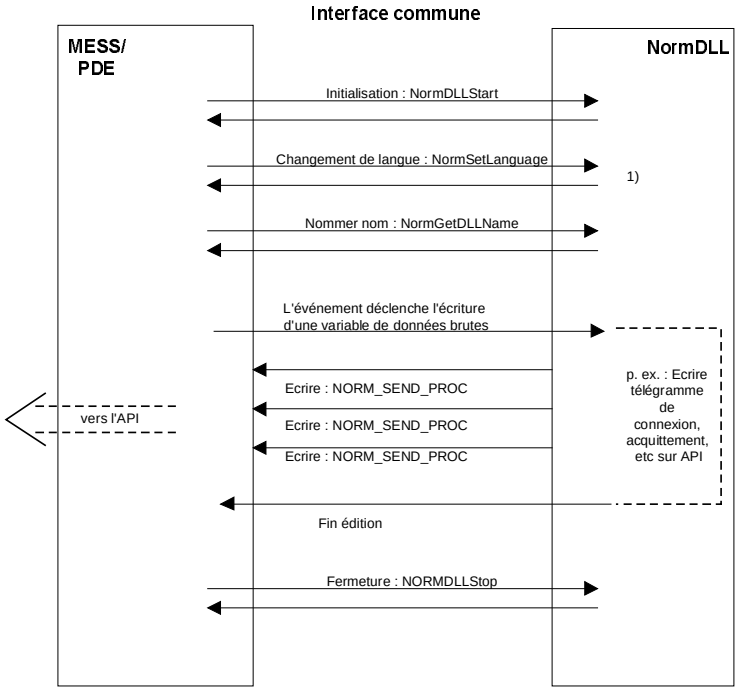
- OS vers automate : (Emission de contrats de connexion/déconnexion, acquittements)
- Automate vers OS : (Réception d'alarmes et de données d'archive)

Par un appel d'initialisation, TagLogging/AlarmLogging communiquent à la DLL S7PMC-DLL les noms des variables d'archive et les numéros d'alarme configurés. Pour ces objets, la DLL de normalisation (WinCC) doit se déclarer à l'automate. L'appel d'initialisation peut être exécuté à n'importe quel moment.

La DLL de normalisation est appelée par AlarmLogging/TagLogging pour la désinitialisation afin de rendre les ressources, etc.

5.3.1 Interface commune vers AlarmLogging et TagLogging

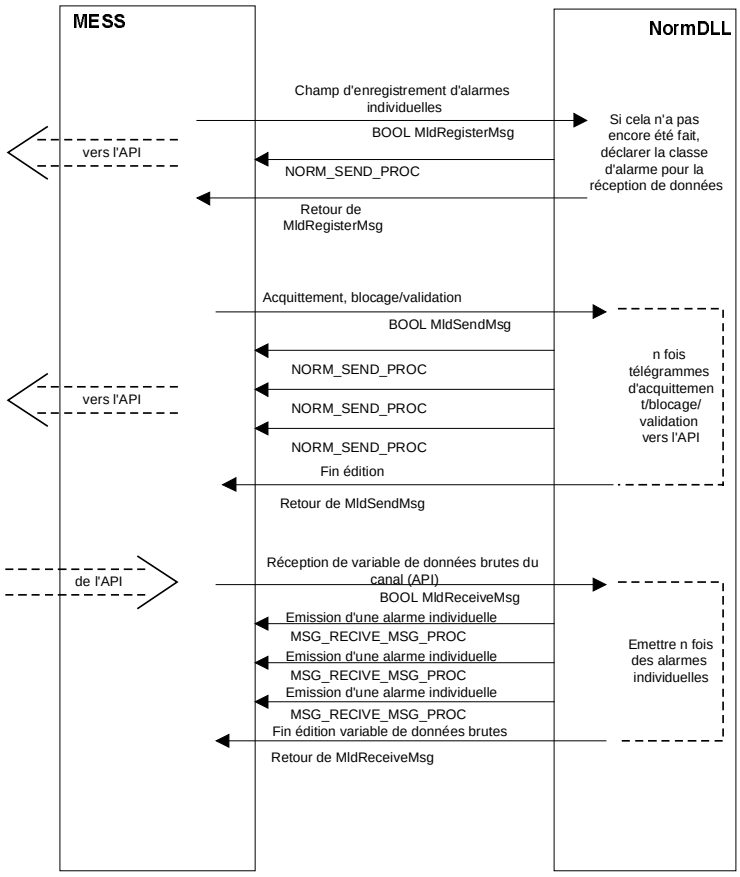
Les fonctions générales de la DLL de normalisation, qui sont identiques pour AlarmLogging et TagLogging, sont regroupées dans une interface commune. Les noms de fonctions commencent tous par 'NORM...'.
(Préfixe des fonctions spécifiques à AlarmLogging : 'Mld...'; préfixe des fonctions spécifiques à TagLogging : 'Pde...'.)



1) Changement de langue uniquement nécessaire pour les DLL de normalisation possédant un dialogue

MELD = AlarmLogging
PDE = TagLogging

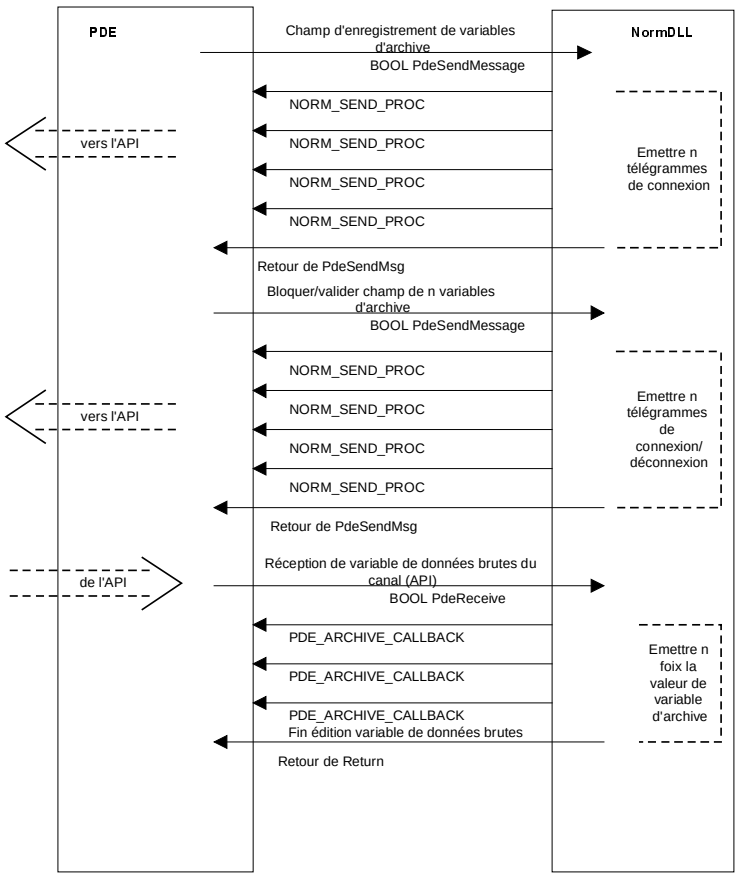
Complément spécifiques à AlarmLogging
Runtime



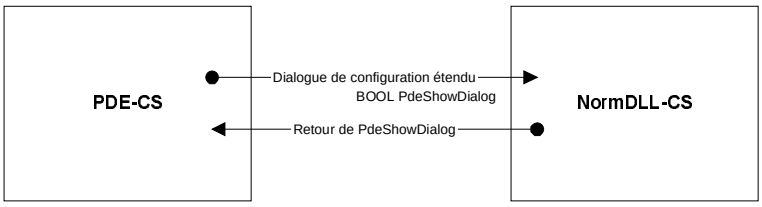
Dialogue de configuration étendu

5.3.2 Compléments spécifiques à TagLogging

Runtime



Dialogue de configuration étendu



5.3.3 Fonctions API d'une DLL de normalisation WinCC

La DLL de normalisation est constituée des domaines ci-dessous:

- Initialisation de la DLL de normalisation
  - Initialisation par le système d'exploitation pendant le chargement de la DLL de normalisation (LibMain)
  - Interrogation des propriétés d'une DLL de normalisation
  - Interrogation du nom de la DLL de normalisation
- Fermeture de la DLL de normalisation
  - Fermeture par TagLogging et AlarmLogging
  - Déchargement par le système d'exploitation
- Extension de la configuration
  - Extension du dialogue de configuration des alarmes
  - Extension du dialogue de configuration des variables d'archive
- Services en ligne
  - Enregistrement de tous les objets spécifiques à la DLL de normalisation (alarmes, variables d'archive)
  - Changement de langue
- Normalisation
  - Normalisation d'alarmes
  - Normalisation de variables d'archive

5.3.3.1 Initialisation de la DLL de normalisation

Initialisation pendant le chargement

Les applications AlarmLogging et/ou TagLogging chargent une DLL de normalisation WinCC avec l'appel système *LoadLibrary*. La DLL de normalisation est alors chargée par le système d'exploitation et initialisée par les mécanismes standard du SE. Toutes les adresses d'entrée de la DLL de normalisation sont définies.

5.3.3.2 Interrogation des propriétés d'une DLL de normalisation

Avec l'appel NormDLLStart (voir ci-dessous), AlarmLogging et TagLogging se déclarent à la DLL de normalisation concernée. Cet appel est destiné à permettre l'échange d'informations entre la DLL de normalisation et l'application.

NormDLLStart

```
include <winccnrm.h>

BOOL NormDLLStart(
        LPVOID lpUser,
        BOOL bModeRuntime,
        PNORM_STARTSTRUCT pcis,
        PCMN_ERROR lpError);
```

Paramètres	Description
lpUser	Pointeur sur données d'application ; à passer sans modification à Callback
bModeRuntime	TRUE lorsque DLL de normalisation est lancée en runtime; FALSE en mode configuration ; pas exploité actuellement par DLL de normalisation.
pcis	Pointeur sur structure de démarrage.
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Aucune erreur
FALSE	Erreur dans fonction API ; description de la cause de l'erreur via le pointeur lpError

NORM_STARTSTRUCT

Paramètres	Description	E/S
dwSize	Taille de la structure en octets	S
lpstrProjectPath	Chemin du projet courant sélectionné	E
NORM_SEND_PROC pfnWriteRwData	Pointeur sur fonction Callback de l'application par laquelle la DLL de normalisation envoie une variable de données à l'automate via le gestionnaire de données.	E
dwAppID	Code application: 1 = AlarmLogging 2 = TagLogging 3 = USER (réservé pour autres applications ; pas utilisé actuellement)	E
dwLocaleID	Langue courante paramétrée au moment de l'appel	E
dwNormCap	Propriétés de la DLL de normalisation d'après tableau suivant	S

La fonction Callback servant à transmettre des variables de données brutes au gestionnaire de données WinCC est alimentée comme suit:

```
ypedef BOOL(*NORM_SEND_PROC)(
        LPDM_VAR_UPDATE_STRUCT    lpDmVarUpdate,
        DWORD                     dwWait,
              LPVOID              lpUser,
        LPCMN_ERROR               lpError );
```

Paramètres	Description
lpDmVarUpdate	Pointeur sur variable de données brutes
dwWait	Code indiquant si l'application doit attendre la fin de l'appel pour écriture ou pas: WAIT_ID_NO avec SET_VALUE WAIT_ID_YES avec SET_VALUE_WAIT
lpUser	Pointeur sur données d'application ; relevé à l'appel de NormDLLStart
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Aucune erreur
FALSE	Erreur dans fonction API ; description de la cause de l'erreur via le pointeur lpError

Un bit est affecté à chaque propriété selon le tableau suivant.

Propriété	Masque de bits		Signification
NORMCAP_DIALOG	0x00000001	activée	La DLL de normalisation propose un dialogue spécial
		désactivée	La DLL de normalisation ne propose aucun dialogue
NORMCAP_REENTRANT	0x00000002	activée	La DLL de normalisation est réentrante
		désactivée	La DLL de normalisation n'est pas réentrante
NORMCAP_MSG_FREE_LOCK	0x00000004	activée	Connexion/déconnexion pour alarmes possible.
		désactivée	Connexion/déconnexion pour alarmes pas possible.
NORMCAP_ARC_FREE_LOCK	0x00000008	activée	Connexion/déconnexion pour variables d'archive possible.
		désactivée	Connexion/déconnexion pour variables d'archive pas possible.
NORMCAP_MSG_GENERIC	0x00000010	activée	Les alarmes créées peuvent être génériques.
		désactivée	Les alarmes ne peuvent pas être génériques.

Propriété	Masque de bits		Signification
NORMCAP_ARC_GENERIC	0x00000020	activée	Les variables d'archive créées peuvent être génériques.
		désactivée	Les variables d'archive ne peuvent être génériques.

5.3.3.3 Interrogation du nom de la DLL de normalisation

NormGetDLLName

```
include <winccnrm.h>

LPTSTR NormGetDLLName( void );
```

Valeur retournée	Description
LPTSTR	Pointeur sur une chaîne de caractères contenant en clair le nom de la DLL de normalisation ; le nom dépend de la langue courante paramétrée.

5.3.4 Fermeture de la DLL de normalisation

Fermeture par TagLogging et AlarmLogging

AlarmLogging et TagLogging préviennent la DLL de normalisation lorsque les applications sont terminées. Dans la DLL de normalisation, les ressources sont alors rendues en bonne et due forme.

NormDLLStop

```
include <winccnrm.h>

BOOL NormDLLStop (void);
```

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API

Déchargement par le système d'exploitation

Aucune manipulation spéciale n'est nécessaire.

5.3.4.1 Extensions de la configuration

Des indications spécifiques sont nécessaires pour les objets S7PMC. Ces indications sont d'abord demandées dans un dialogue avec des ressources standard (sans MFC) et sont intégrées directement dans le numéro d'alarme WinCC ou dans le nom de la variable d'archive. Ainsi, la DLL de normalisation n'a pas à stocker et gérer elle-même ces indications. Pour garantir l'unicité d'un numéro d'alarme ou d'une variable d'archive dans le projet, une correspondance doit exister entre le numéro d'alarme ou de la variable d'archive et la variable de données brutes associée. Cette information de correspondance fait partie intégrante du numéro d'alarme ou du nom de la variable d'archive.

5.3.4.2 Extension du dialogue de configuration d'alarmes S7PMC

La DLL de normalisation comporte une fonction API pour la définition du numéro d'alarme spécifique S7PMC. Cette fonction est appelée par le système de configuration (CS) du système d'alarmes lors du paramétrage d'alarmes individuelles rattachées à une DLL de normalisation S7PMC. La DLL de normalisation S7PMC attribue comme numéro d'alarme le numéro formé de deux parties :

Partie 1 :

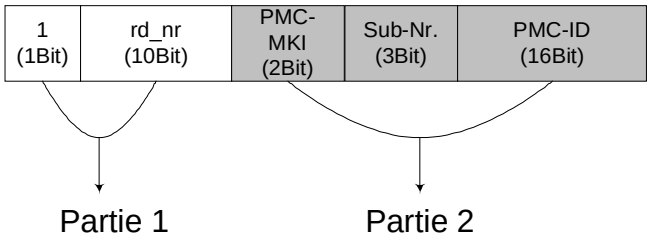
Numéro identifiant de manière univoque une CPU d'automate (numéro de variable de données brutes) dans tout le projet

Partie 2 :

Numéro rattaché à l'alarme et identifiant celle-ci de manière univoque dans une CPU d'automate (spécifique à la DLL de normalisation)

Dans le dialogue de configuration, la sélection suivante doit être faite pour constituer le numéro d'alarme :

Structure d'un numéro d'alarme S7PMC (32 bits)



Partie 1 :

Chaque alarme est associée à une variable de données brutes identifiant une CPU AG. Les conventions ci-dessous ont été définies pour réaliser la mise en correspondance entre variables de données brutes et numéro d'alarme.
Le nom de la variable de données brutes pour S7PMC - et pour tous les modes de liaisons avec DLL de normalisation - présente la structure suivante :

@rd_alarm#rd_nr

@rd_alarm#	Partie fixe du nom d'une variable de données brutes pour DLL de normalisation
rd_nr	Nombre décimal entre 0 et 1023 identifiant une variable de données brutes (sans zéros de tête)

Le bit de plus fort poids du numéro d'alarme est mis à 1 pour les numéros d'alarmes attribués (en externe) par les DLL de normalisation. Ces alarmes ne doivent être traitées que par les DLL de normalisation correspondantes : le numéro d'alarme ne peut être modifié dans le dialogue de configuration d'AlarmLogging.

Partie 2 :

Cette partie du numéro d'alarme peut être attribuée par la DLL de normalisation concernée. Elle a pour S7PMC la signification suivante :

MKl	Classe d'alarmes ; sélectionnez une des classes ci-dessous : SCAN (1) ALARM/NOTIFY (2) ALARM_8P/ALARM_8 (2) LTM (3)
Sous-n°	Sous-numéro d'alarme ; pertinent uniquement pour ALARM_8 et ALARM_8P : 1...8
ID PMC	Numéro d'alarme PMC (paramètre d'entrée bloc EV-ID) : 1...16386 pour classes d'alarmes SCAN et ALARM/NOTIFY ou ALARM_8P/ALARM_8 1...7 pour classe d'alarme LTM

MldShowDialog

```
include <winccnrm.h>

BOOL WINAPI MldShowDialog(
              HWND                  hwnd,
              LPMSG_CSDATA_GENERIC  lpmCS,
              LPDM_PROJECT_INFO     lpDMProjectInfo,
              LPCMN_ERROR             lpError );
```

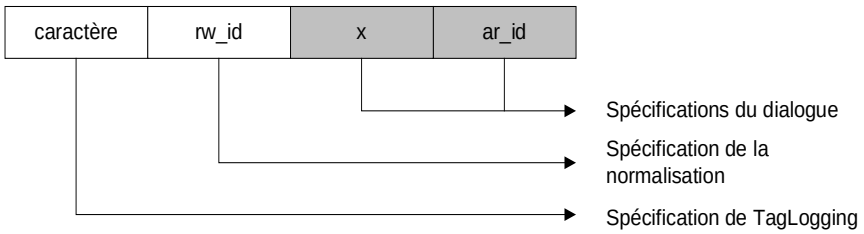
Paramètres	Description
hwnd	Poignée de fenêtre
lpmCS	Pointeur sur données d'alarmes individuelles
lpDMProjectInfo	Pointeur sur infostructure de projet
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

5.3.4.3 Extension du dialogue de configuration de variables d'archive

La DLL de normalisation comporte une fonction API pour la définition des noms de variables d'archive spécifiques à S7PMC. Cette fonction est appelée par le système de configuration (CS) de TagLogging lors du paramétrage des variables d'archive rattachées à une liaison S7PMC. La DLL de normalisation S7PMC attribue comme nom de variable d'archive un nom constitué de plusieurs éléments ; il contient entre autres le numéro rattaché à l'archive côté automate . Par cet algorithme, les numéros d'archive S7PMC sont uniques dans le système de description de variables d'archive WinCC, ce qui permet une mise en correspondance rapide en runtime.
TagLogging garantit l'unicité globale des noms de variables d'archive.

Structure d'un nom de variable d'archive S7PMC (longueur maxi. 18 octets)



Désignation	Longueur en octets	attribué en dialogue	Signification
caractères	9	par TagLogging	Séquence fixe de caractères attribuée par TagLogging ; constituée du nom de la DLL de normalisation et du signe # comme séparateur. p. ex. pour S7PMC: NRMS7PMC ; n'apparaît pas à l'utilisation.
rw_id	8	TagLogging/ Norm.DLL	Identifiant de données brutes en caractères hexadécimaux (y compris zéros de tête) ; permet une affectation univoque à la variable

Désignation	Longueur en octets	attribué en dialogue	Signification
			de données brutes (liaison) à laquelle le numéro d'archive est rattaché. La partie 'nom' est formée par la DLL Norm.DLL à l'aide des paramètres d'entrée de TagLogging.
x	1	DLL de norm. Partie CS	Code spécifique S7PMC pour distinction entre BSEND et AR_SEND : 'A' = AR_SEND 'B' = BSEND
ar_id	4	DLL de norm. Partie CS	Identifiant en caractères hexadécimaux (y compris zéros de tête) Dépend du code x : Numéro d'archive spécifique S7PMC AR_ID ou R_ID spécifique S7 pour BSEND

Exemple d'un nom de variable d'archive créé pour S7PMC : #00000001#A#0014

PdeShowDialog

```
include <winccnrm.h>

BOOL WINAPI PdeShowDialog(
              LPVOID        hwnd,
              LPTSTR        lpszArcVarName,
              DWORD         dwArcVarNameLength,
              LPDM_VARKEY   lpVarKey,
              LPCMN_ERROR   lpError
              );
```

Paramètres	Description
hwnd	Poignée de fenêtre
lpszArcVarName	Pointeur sur champ de chaîne de caractères pour stockage de la partie 'nom' de la variable d'archive spécifique à la DLL de normalisation
dwArcVarNameLength	Longueur maxi de la partie 'nom' spécifique à la DLL de normalisation
lpVarKey	Pointeur sur Varkey de la variable de données brutes
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

5.3.4.4 Services en ligne

Enregistrer toutes les alarmes

Cette fonction est nécessaire parce que la DLL de normalisation ne contient pas d'informations de configuration sur les alarmes pertinentes. Les alarmes ne sont envoyées par l'automate que lorsque l'application (WinCC) s'est déclarée prête à recevoir les alarmes. AlarmLogging appelle la fonction MldRegisterMsg à l'heure actuelle pour chaque alarme pertinente et transmet ainsi à la DLL de normalisation les informations de configuration de l'alarme individuelle. Outre la description de l'alarme, la DLL de normalisation reçoit un pointeur sur la variable de données brutes (liaison) affectée à cette alarme. Au runtime, la DLL de normalisation peut ainsi créer dans la mémoire vive un tableau permettant de constituer des télégrammes d'alarmes spécifiques S7PMC.

MldRegisterMsg

```
include <winccnrm.h>

BOOL WINAPI MldRegisterMsg(
              LPDM_VARKEY   lpDMVarKey,
              LPDWORD       lpMsgNumber,
              DWORD         DwNumMsgNumber,
              LPCMN_ERROR   lpError );
```

Paramètres	Description
lpDMVarKey	Pointeur sur Varkey de la variable de données brutes
lpMsgNumber	Pointeur sur champ de numéros d'alarmes individuelles
dwNumMsgNumber	Nombre de numéros d'alarmes individuelles
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

5.3.4.5 Enregistrer toutes les variables d'archive

Cette fonction est nécessaire parce que la DLL de normalisation ne contient pas d'informations de configuration sur les variables d'archive pertinentes. C'est pourquoi la fonction PdeSendMsg est appelée pour un certain nombre de variables d'archive pertinentes ; ainsi les informations de configuration et les informations complémentaires TagLogging pour les variables d'archive sont notifiées.
Il est possible d'enregistrer plusieurs variables d'archive d'une liaison avec un seul appel.
Pour chaque variable d'archive, TagLogging transmet un double mot d'informations complémentaires à la DLL de normalisation, ces informations étant gérées dans la mémoire de la DLL de normalisation. TagLogging a besoin de ces informations complémentaires dès que des valeurs de variables d'archive doivent être traitées (dans la fonction Callback TagLogging_ARCHIVE_CALLBACK).
Ainsi la DLL de normalisation peut au runtime créer dans la mémoire vive un tableau permettant de constituer les télégrammes d'alarme spécifiques S7PMC pour l'archive concernée. Les télégrammes d'alarmes sont nécessaires pour signaler à l'automate l'état 'prêt à recevoir' pour le numéro d'archive concerné. Ce n'est qu'après une déclaration réussie que l'automate envoie les données d'archive à l'application (WinCC).

PdeSendMsg

```
include <winccnrm.h>

BOOL WINAPI PdeSendMsg(
              NORM_SEND_PROC      lpfnCallBack,
              DWORD               dwFunctionId,
              LPSZ_ARC_VAR_NAME   lpszArcVarName,
              LPDWORD             lpdwData
              DWORD               dwNumArchVarName,
              LPDM_VARKEY         lpVarKey,
              LPVOID              lpUser,
              LPCMN_ERROR         lpError
              );
```

Paramètres	Description
lpfnCallBack	Pointeur sur routine Callback devant transmettre au gestionnaire de données la variable de données brutes formée par la DLL de normalisation. Si la valeur est zéro, la routine Callback est appelée par la structure Ini. L'adresse de fonction fournie par la structure Ini n'est pas identique à ce paramètre.
dwFunctionId	Code de fonction FUNC_ID_REGISTER (voir tableau ci-dessous) ; la même fonction est valable pour toutes les variables mentionnées
lpszArcVarName	Pointeur sur un tableau de pointeurs dont les éléments renvoient aux noms des variables d'archive
lpdwData	Pointeur sur un champ dont les éléments contiennent des données complémentaires pour les variables d'archive ; peut être zéro. A l'exécution de la fonction FUNC_ID_REGISTER (déclaration de variable d'archive) la valeur complémentaire rattachée à une variable d'archive est reprise inchangée dans des listes internes de la DLL de normalisation et passée en temps utile à TagLogging_ARCHIVE_CALLBACK. Sans utilité pour les autres codes de fonction.
dwNumArchVarName	Nombre des noms de variable d'archive à traiter
lpVarKey	Pointeur sur Varkey de la variable de données brutes
lpUser	Pointeur sur données utilisateur ; à passer sans modification à Callback
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

Fonctions possibles de la procédure PdeSendMsg (valeurs de dwFunctionId) :

Propriété	Masque de bits	Signification
FUNC_ID_LOCK	0x00000001	Verrouiller variable d'archive
FUNC_ID_FREE	0x00000002	Autoriser variable d'archive

Propriété	Masque de bits	Signification
FUNC_ID_ REGISTER	0x00000004	Déclarer variable d'archive
FUNC_ID_ UNREGISTER	0x00000008	Déclarer suppression variable d'archive dans enregistrement (pas utilisé actuellement)

5.3.4.6 Commutation de langue

Le dialogue de configuration doit être spécifique à la langue : la DLL de normalisation doit savoir quelle langue est paramétrée. Le paramétrage de la langue est transmis dans la structure de démarrage au lancement. La commutation dynamique de langues doit être également passée par TagLogging et AlarmLogging à la DLL de normalisation. Ceci se fait avec l'appel

NormSetLanguage

```
include <winccnrm.h>

BOOL NormSetLanguage(
              DWORD            dwLocaleID,
              LPCMN_ERROR         lpError
              );
```

Paramètres	Description
dwLocaleID	Langue courante paramétrée au moment de l'appel
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

5.3.5 Normalisation

Lorsqu'une application s'est déclarée à l'automate 'prête à recevoir' des alarmes ou des données d'archive, elle reçoit les données via la variable de données brutes correspondante. La déclaration est effectuée par la DLL de normalisation lors de l'enregistrement. A partir de ce moment, des télégrammes de données peuvent être reçus de l'AS. Les télégrammes de données sont incorporés dans des variables de données brutes et transmis, par la DLL de canal, le gestionnaire de données et l'application correspondante (en l'occurrence TagLogging ou AlarmLogging), à la DLL de normalisation compétente pour ce type de variables de données brutes. La DLL de normalisation interprète les données qui arrivent et génère à partir de celles-ci les alarmes ou les données d'archive.

5.3.5.1 Dérivation d'alarmes individuelles

Une variable de données brutes (d'un télégramme) permet de stocker *n* alarmes individuelles. La DLL de normalisation doit interpréter ce télégramme spécifique S7PMC et transmettre à AlarmLogging les alarmes individuelles en résultant.

Le numéro d'alarme (EV_ID) de S7PMC est un constituant du numéro d'alarme WinCC.
Une alarme permet à S7PMC de fournir jusqu'à 10 valeurs de process. Le type de données «string» est également admis ici comme valeur de process. Ce type de valeur de process n'est pas supporté par AlarmLogging ; les valeurs complémentaires de ce type doivent être rejetées par la DLL de normalisation.

La fonction MldReceiveMsg est aussi appelée par AlarmLogging chaque fois que l'état de la variable de données brutes est modifié, c'est-à-dire lorsque l'état de dérangement est constaté après l'état OK ou inversement par le gestionnaire de données. Le changement d'état de la variable de données brutes (qui correspond à une liaison) est important pour la DLL de normalisation S7PMC. Pour de plus amples informations, voir chapitre «Exécution sur changement d'état».

MldReceiveMsg

```
include <winccnrm.h>

BOOL WINAPI MldReceiveMsg(
              MSG_RECEIVE_MSG_PROC    lpfnMsgReceive,
              LPDM_VAR_UPDATE_STRUCT  lpDMVar,
              LPVOID                  lpUser,
              LPCMN_ERROR             lpError );
```

Paramètres	Description
lpfnMsgReceive	Pointeur sur routine Callback devant transmettre à AlarmLogging l'alarme individuelle formée par la DLL de normalisation.
lpDMVar	Pointeur sur variable de données brutes
lpUser	Pointeur sur données utilisateur ; à passer sans modification à Callback
lpError	Pointeur sur structure d'erreurs standard WinCC
Valeur retournée	**Description**
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

La fonction Callback servant à l'envoi des alarmes individuelles à AlarmLogging est alimentée comme suit :

```
typedef BOOL(*MSG_RECEIVE_MSG_PROC)(
              LPMSG_RTCREATE_STRUCT   lpMsgCreate,
              DWORD                   dwNumMsg,
              LPVOID                  lpUser,
              LPCMN_ERROR             lpError );
```

Paramètres	Description
lpMsgCreate	Pointeur sur alarme WinCC
dwNumMsg	Nombre d'alarmes individuelles
lpUser	Pointeur sur données d'application
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

5.3.5.2 Acquittement, verrouillage/autorisation d'alarmes

Le concept d'alarmes et d'acquittement d'alarmes de WinCC-AlarmLogging et S7PMC prévoit que les alarmes peuvent être acquittées en fonction de leur configuration. L'information d'acquittement est connue d'AlarmLogging, mais doit être également gérée dans la mémoire d'acquittements d'alarmes de l'automate. Pour cela, AlarmLogging envoie des télégrammes d'acquittement à l'automate par la DLL de normalisation associée à la liaison.
A partir des données d'entrée, la DLL de normalisation S7PMC construit des télégrammes S7PMC correspondants qui seront transmis au gestionnaire de données par la fonction Callback NORM_SEND_PROCd'AlarmLogging.
La même procédure s'applique pour le verrouillage/la réautorisation d'une alarme individuelle par AlarmLogging : la génération de l'alarme est inhibée/réautorisée à la source dans l'automate.

MldSendMsg

```
include <winccnrm.h>

BOOL WINAPI MldSendMsg(
              NORM_SEND_PROC          lpfnMsgSend,
              LPMSG_SEND_DATA_STRUCT  lpSendData,
              DWORD                   dwNumData,
              LPVOID                  lpUser,
              LPCMN_ERROR             lpError );
```

Paramètres	Description
lpfnMsgSend	Pointeur sur routine Callback d'AlarmLogging avec laquelle la variable de données brutes formée par la DLL de normalisation doit être transmise à l'automate pour écriture. Les paramètres sont décrits au chapitre «Interrogation des propriétés d'une DLL de normalisation».
lpSendData	Pointeur sur données d'émission ; structure décrite ci-dessous
dwNumData	Nombre des contrats individuels à traiter
lpUser	Pointeur sur données utilisateur ; à passer sans modification à Callback
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

Structure des données d'émission AlarmLogging (contrat individuel)

Variable	Signification
DWORD dwVarID	Variables de données brutes - Identifiant du GD
DWORD dwNotify	Valeurs possibles de Notify MSG_STATE_QUIT Acquittement alarme MSG_STATE_LOCK Verrouillage alarme MSG_STATE_UNLOCK Autorisation alarme MSG_STATE_QUIT_EMERGENCY Acquitter toutes les alarmes
DWORD dwData	si QUIT, LOCK, UNLOCK --> numéro d'alarme si NOTQUIT --> pas utilisé

5.3.5.3 Exécution sur changement d'état

Le changement d'état d'une liaison (variable de données brutes) doit être communiqué à la DLL de normalisation. Ceci est effectué avec la fonction **MldReceiveMsg**.

Changement d'état de - à	Edition dans la DLL de normalisation S7PMC
Défaut - OK	Envoyer télégrammes de déclaration pour toutes les classes d'alarmes S7PMC pour lesquelles une alarme au moins a été configurée à l'automate. La déclaration est effectuée de manière spécifique aux classes d'alarmes S7PMC. La DLL de normalisation connaît déjà toutes les alarmes configurées en raison de leur enregistrement.
OK - Défaut	La DLL de normalisation doit refuser les contrats actifs déjà envoyés à l'automate, mais qui ne peuvent plus être traités complètement en raison du changement d'état (des acquittements manquent).

5.3.5.4 Mise à jour des alarmes de la DLL de normalisation S7PMC

Lors de la mise à jour, la DLL de normalisation S7PMC lit l'état de toutes les alarmes dont elle a connaissance par l'enregistrement et le transmet comme alarme individuelle à AlarmLogging. Ceci garantit une vue d'alarmes homogène au démarrage du système.

Une mise à jour d'alarmes est nécessaire lorsque

- un changement d'état 'Défaut vers OK' a été détecté (également le cas implicitement au démarrage du système)
- L'automate envoie un «télégramme de mise à jour d'alarmes». Ce télégramme est émis à toutes les stations déclarées lorsque, par exemple, un débordement d'alarmes est constaté lorsque des alarmes sont acquittées ou autorisées par d'autres stations.

Lors de la mise à jour d'alarmes, l'automate envoie les états d'acquittement d'alarmes et les codes de verrouillage. Les valeurs complémentaires et l'heure des alarmes ne sont pas envoyées. La DLL de normalisation fournit alors l'heure système courante comme heure de l'alarme individuelle et traite avec l'opérateur logique OU le code MSG_STATE_UPDATE dans l'état d'alarme.

5.3.5.5 Normalisation de variables d'archive

La DLL de normalisation met deux fonctions à la disposition de TagLogging :

- Dérivation de valeurs de variables d'archive individuelles à partir du contenu d'une variable de données brutes
- Verrouillage / autorisation de variables d'archive

5.3.5.6 Dérivation de valeurs de variables d'archive individuelles

Une variable de données brutes (d'un télégramme) permet de stocker n valeurs de variables d'archive. La DLL de normalisation doit interpréter ce télégramme spécifique S7PMC et passer à AlarmLogging les valeurs de variables d'archive en résultant .

Des opérateurs de conversion de valeurs de process peuvent être envoyés pour une variable d'archive. La DLL de normalisation S7PMC effectue alors la conversion souhaitée de la valeur de process en valeur de variable d'archive. Les fonctions d'échelle existant déjà dans WinCC sont alors utilisées. La procédure exacte est encore à définir.

La fonction PdeReceive est aussi appelée par TagLogging chaque fois que l'état de la variable de données brutes est modifié, c'est-à-dire lorsque l'état Défaut est constaté après l'état OK ou inversement par le gestionnaire de données. Le changement d'état de la variable de données brutes (qui correspond à une liaison) est important pour la DLL de normalisation S7PMC. Pour de plus amples informations, voir chapitre «Exécution sur changement d'état».

PdeReceive

```
include <winccnrm.h>

BOOL PdeReceive (
              LPDM_VAR_UPDATE_STRUCT        lpDmVarUpdate,
              TagLogging_ARCHIVE_CALLBACK   lpfnCallBack,
              LPVOID                        lpUser,
              LPCMN_ERROR                   lpError
              );
```

Paramètres	Description
lpDmVarUpdate	Pointeur sur variable de données brutes
lpfnCallBack	Pointeur sur routine Callback avec laquelle la DLL de normalisation passe les valeurs de variable d'archive individuelles à TagLogging.
lpUser	Pointeur sur données utilisateur ; à passer sans modification à Callback
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

La fonction Callback servant à passer les valeurs des variables d'archive individuelles à TagLogging est alimentée comme suit :

```
OOL (*PDE_ARCHIVE_CALLBACK) (
              LPTSTR        lpszArcVarName,
              double        doValue,
              SYSTEMTIME*   lpstTime
              DWORD         dwFlags
              DWORD         dwData,
              LPVOID        lpUser,
              LPCMN_ERROR   lpError
              );
```

Paramètres	Description
lpszArcVarName	Nom de variable d'archive à partir de l'identifiant de données brutes
doValue	Valeur de variable d'archive
lpstTime	Pointeur sur horodatage dérivant la variable de données brutes des données utiles
dwFlags	Codes dont la signification exacte reste encore à définir.
dwData	Données complémentaires transmises à l'enregistrement de la variable d'archive ; à passer inchangées
lpUser	Pointeur sur données utilisateur ; à reprendre inchangé par l'appel de fonction
lpError	Pointeur sur structure d'erreurs standard WinCC

Valeur retournée	Description
TRUE	Fonction OK
FALSE	Erreur dans fonction API ; description de la cause de l'erreur avec le pointeur `lpError`

5.3.5.7 Verrouillage / autorisation de variables d'archive

Avec cette fonction, TagLogging exploite la possibilité offerte par S7PMC, de piloter la réception de valeurs de variables d'archive. La DLL de normalisation S7PMC génère alors un appel pour déconnexion ou reconnexion de l'archive correspondante et transmet cet appel au gestionnaire de données avec NORM_SEND_PROC.
Du point de vue de la DLL de normalisation S7PMC, le verrouillage / l'autorisation de variables d'archive sont presque identiques aux fonctions nécessaires pour l'enregistrement d'une variable d'archive. C'est pourquoi cette fonction est appelée, pour les deux fonctions, dans la DLL de normalisation PdeSendMsg.
Le code de fonction dwFunctionId permet la distinction entre l'enregistrement et la fonction de verrouillage / autorisation : Pour le verrouillage / l'autorisation, les données complémentaires de chaque variable d'archive lpdwData n'ont aucune utilité. Voir chapitre «Enregistrer toutes les variables d'archive»

5.3.5.8 Exécution sur changement d'état

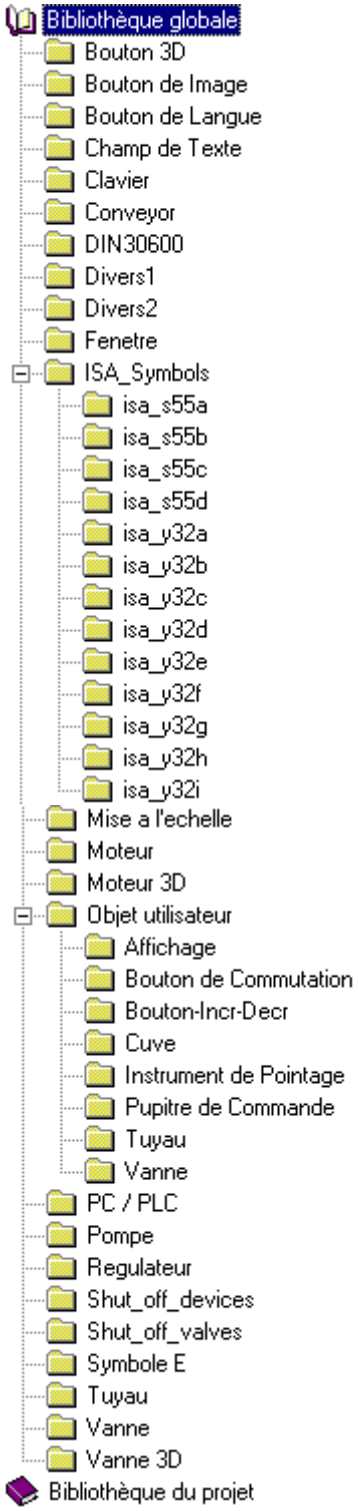
Le changement d'état d'une liaison (variables de données brutes) doit être connu de la DLL de normalisation ; ceci est effectué par la fonction **PdeReceive**.

Changement d'état de - à	Edition dans la DLL de normalisation S7PMC
Défaut - OK	Télégrammes de connexion pour toutes les variables d'archive de toutes les liaisons La DLL de normalisation connaît déjà toutes les variables d'archive configurées en raison de leur enregistrement.
OK - Défaut	La DLL de normalisation doit rejeter les contrats actifs déjà envoyés à l'automate, mais qui ne peuvent plus être traités complètement en raison du changement d'état (des acquittements manquent).

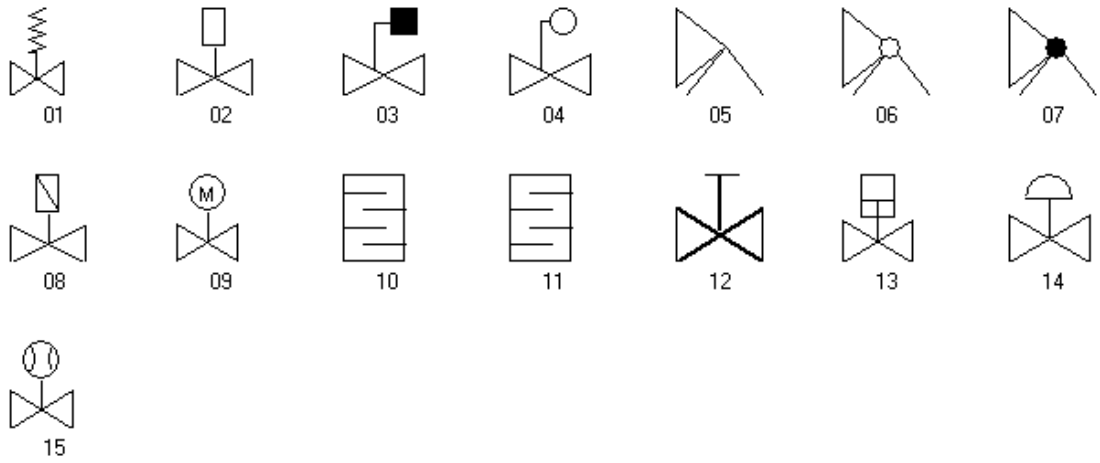
5.4 Composants de vue avec Visual Basic 5.0

Ce chapitre n'existe plus. Il est remplacé par la documentation de l'option WinCC : *IndustrialX.*

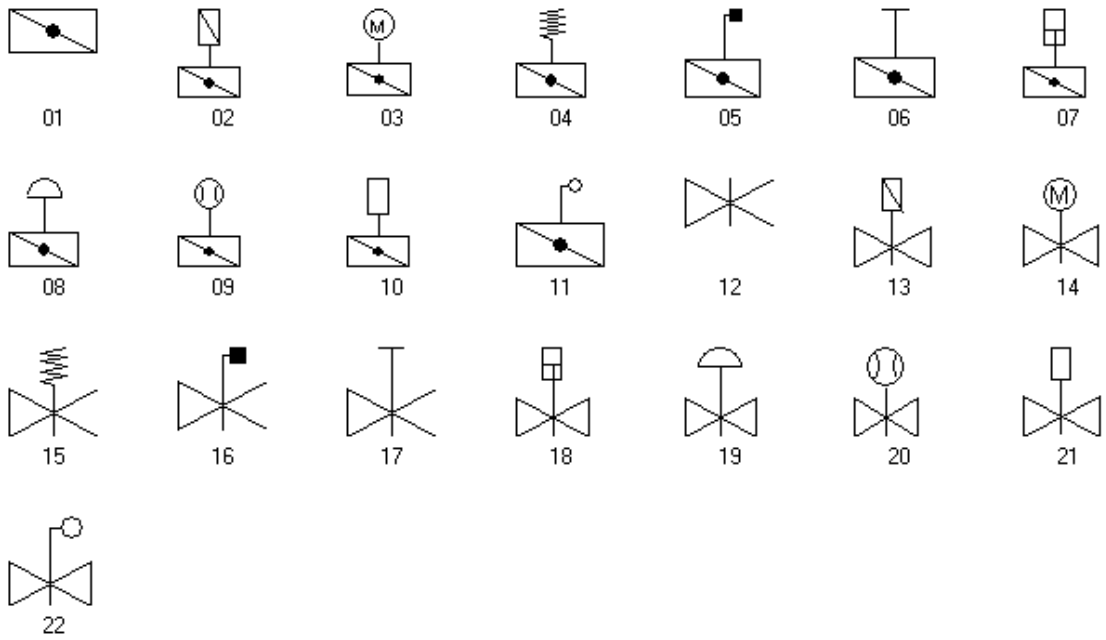
5.5 Bibliothèque globale



5.5.1 Shut off devices



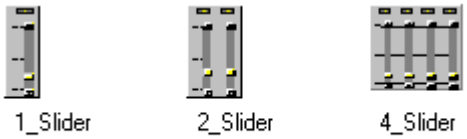
5.5.2 Shut off valves



5.5.3 Objet utilisateur

5.5.3.1 Afficheurs

5.5.3.2 Pupitre de Commande

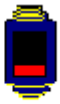


5.5.3.3 Bouton-Incr-Decr

5.5.3.4 Tuyau

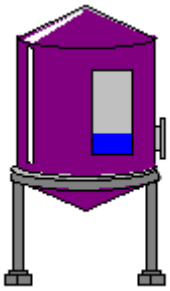
5.5.3.5 Cuve


Cuve 1


Cuve 2


Cuve 3


Cuve 4



5.5.3.6 Bouton de Commutation


In_Out_1


In_Out_2


In_Out_3


In_Out_4


In_Out_5

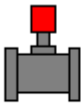

In_Out_6

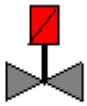

In_Out_7


In_Out_8

5.5.3.7 Vanne


Vanne 1


Vanne 2

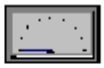

Vanne 3


Vanne 4

5.5.3.8 Instrument de Pointage

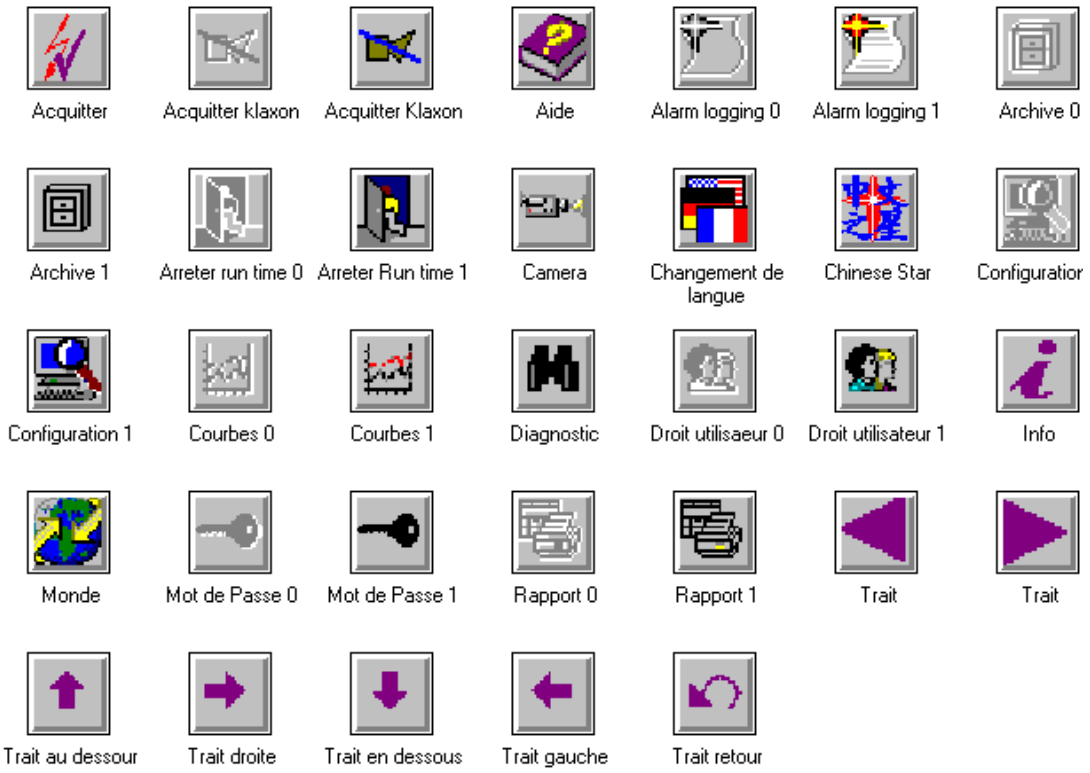

Metre 1_0-100


Metre 2_Min-Max

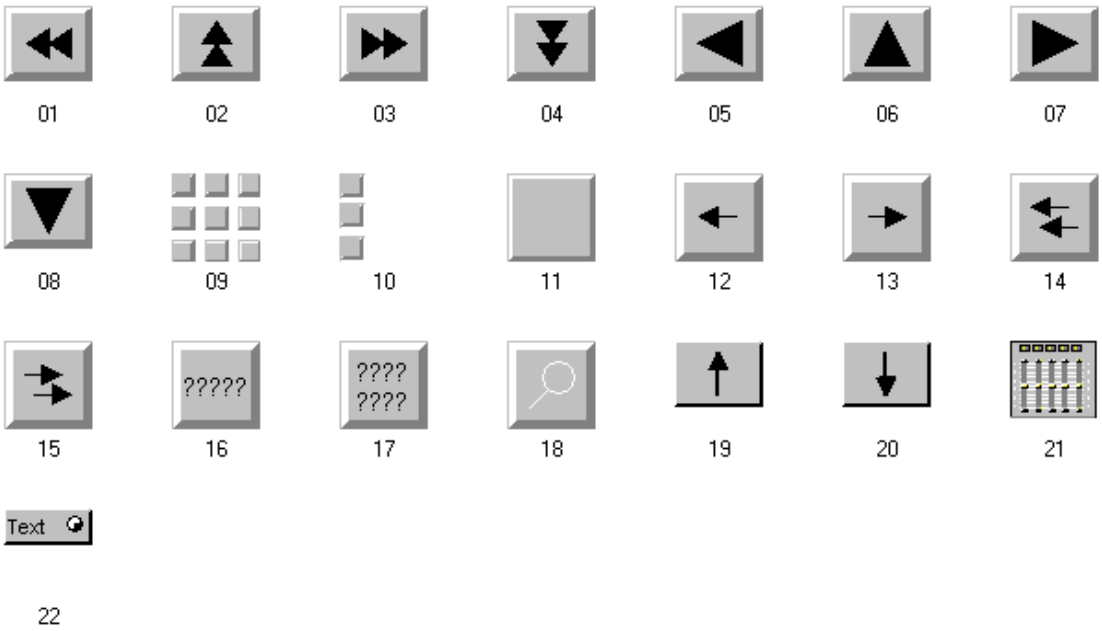

Metre 3_Min-Max


Metre 4_Min-Max

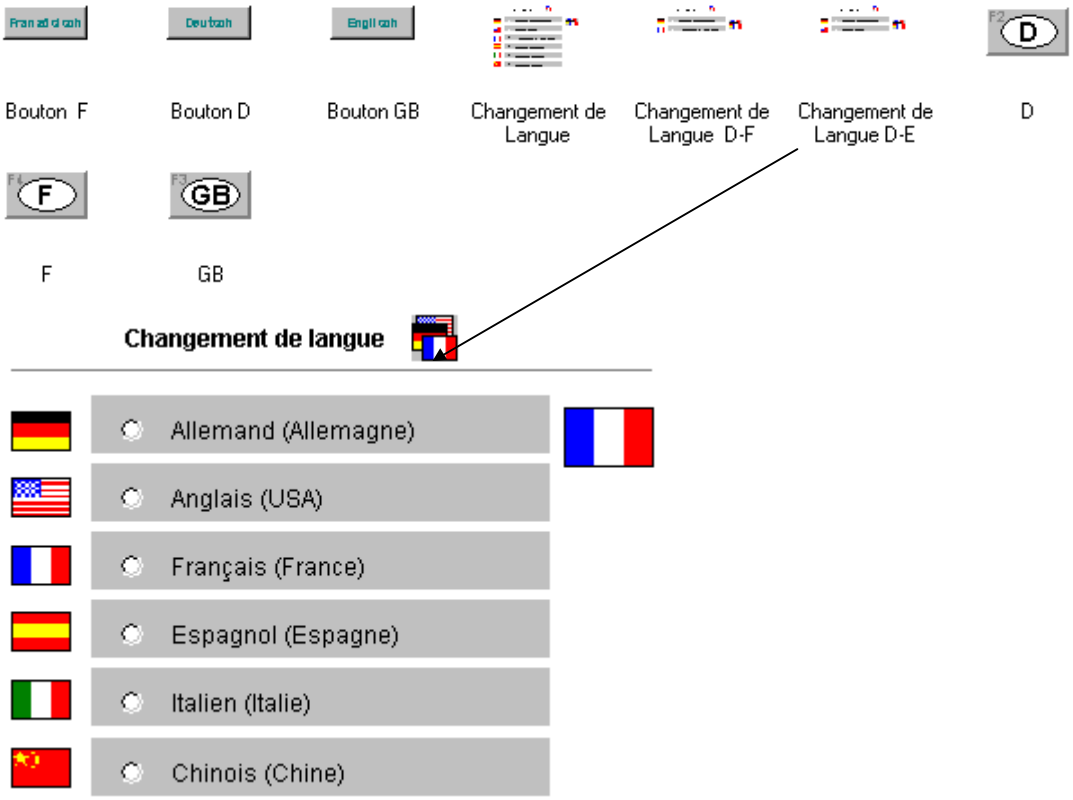
5.5.4 Bouton de Image



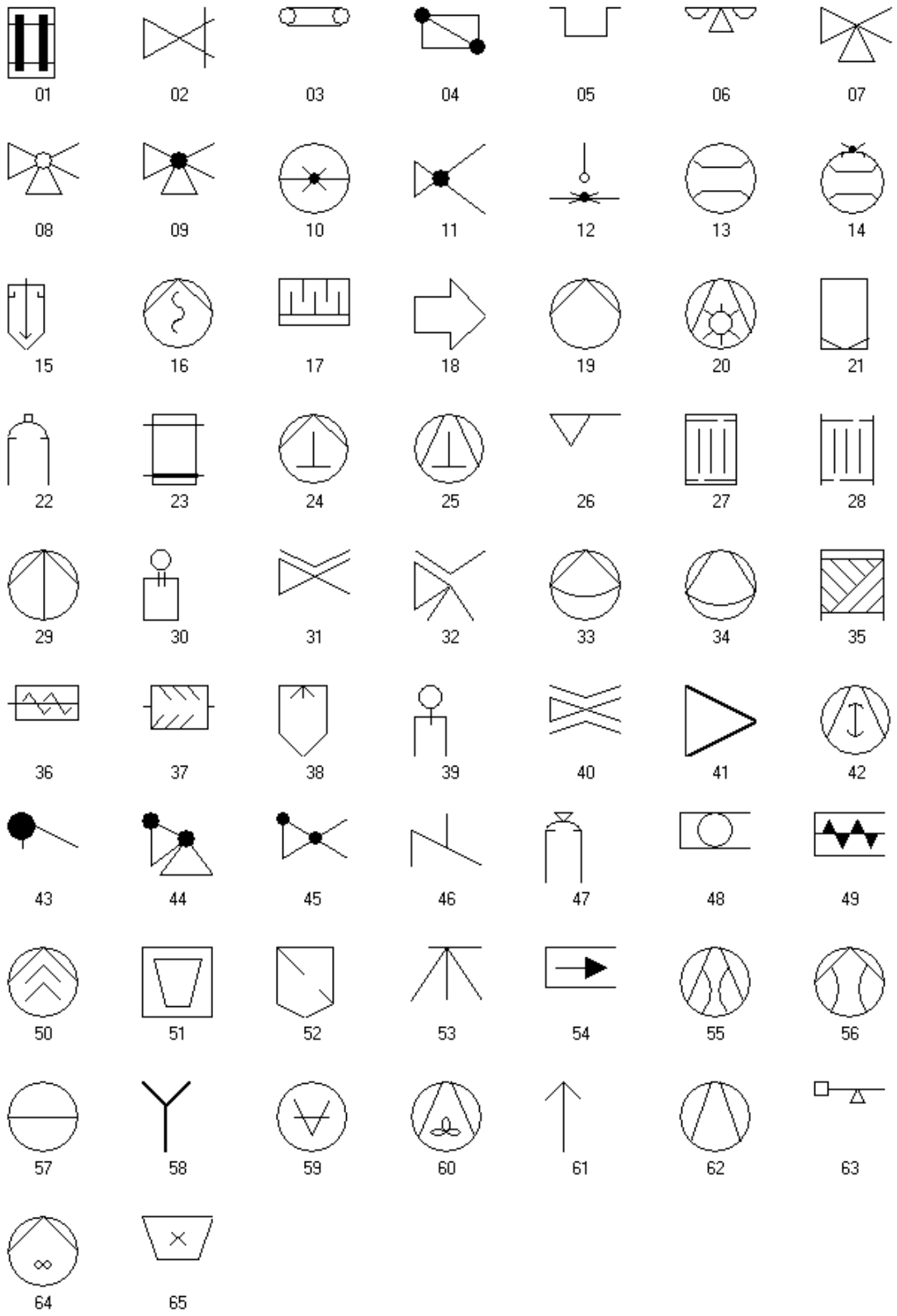
5.5.5 Bouton 3D



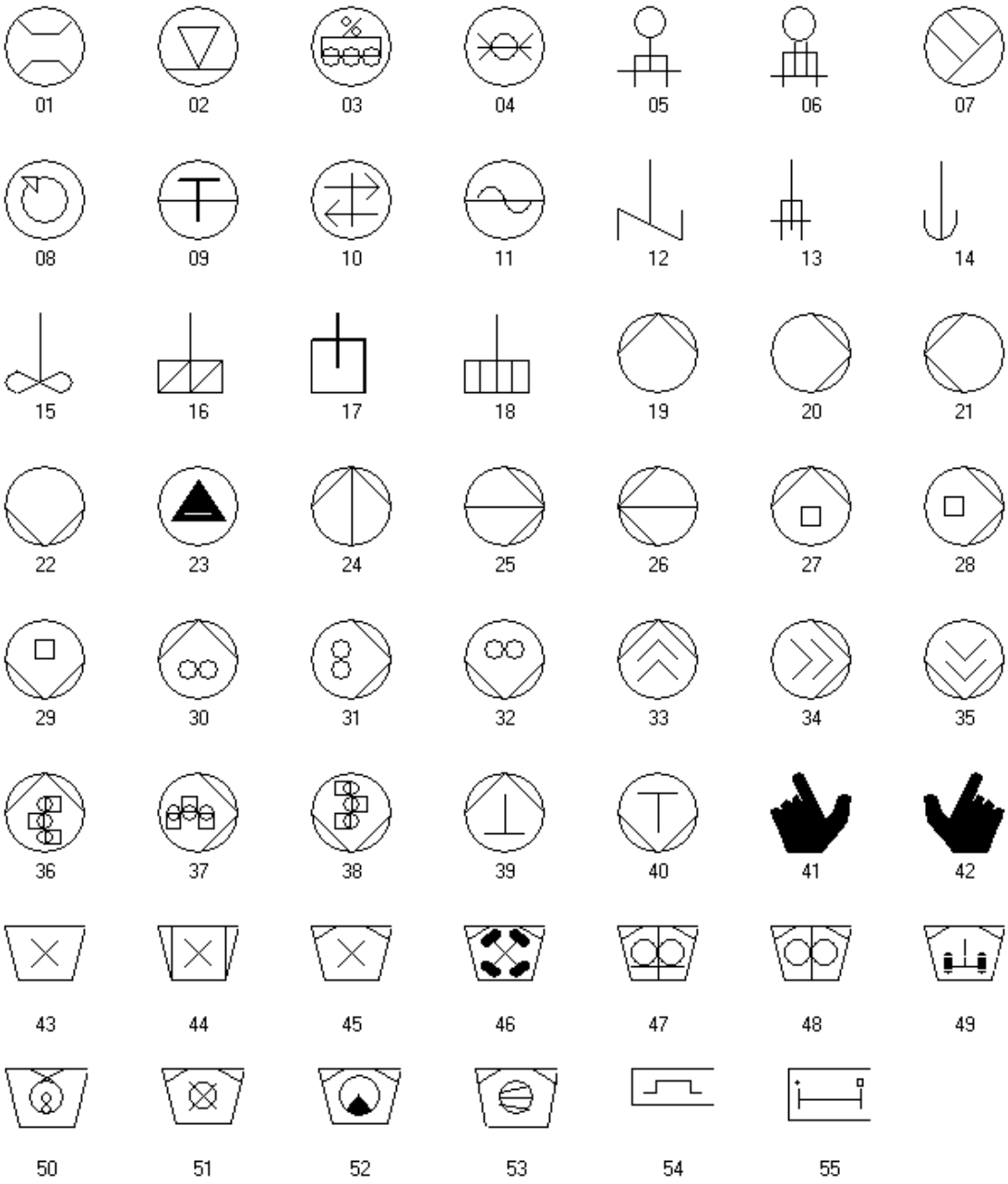
5.5.6 Bouton de Langue



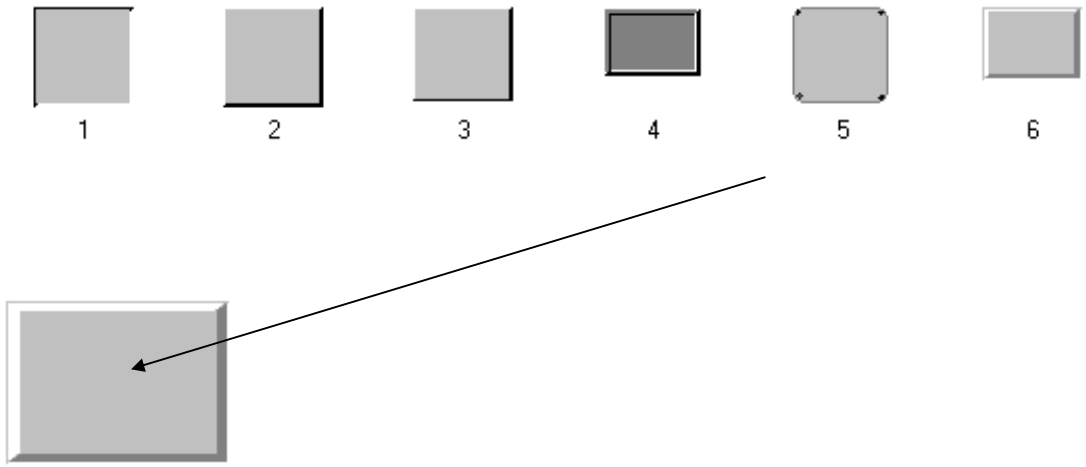
5.5.7 DIN30600



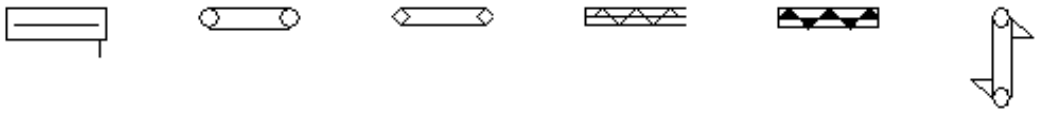
5.5.8 Symbole E



5.5.9 Fenetre

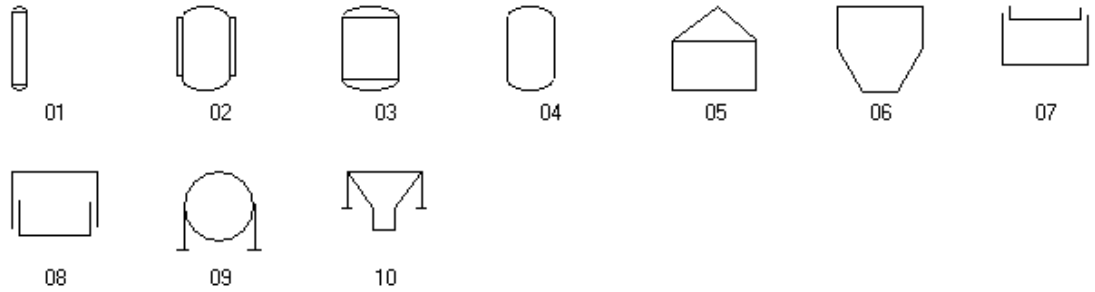


5.5.10 Conveyor

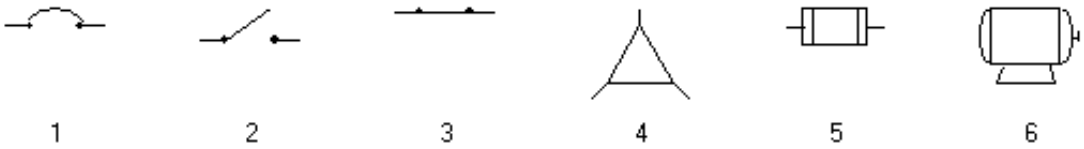


5.5.11 ISA Symbols

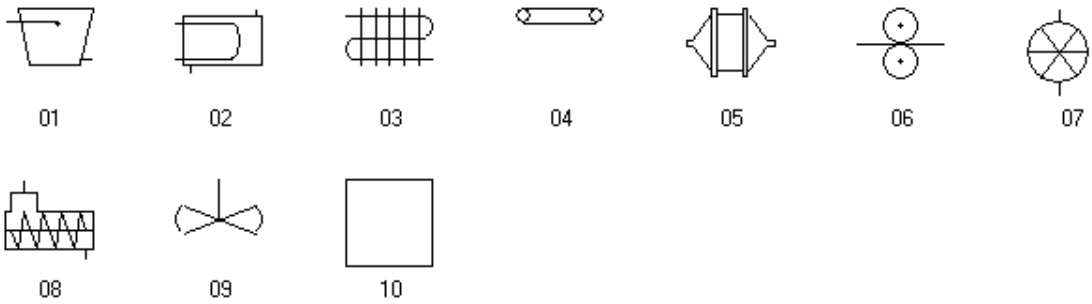
5.5.11.1 isa_s55a



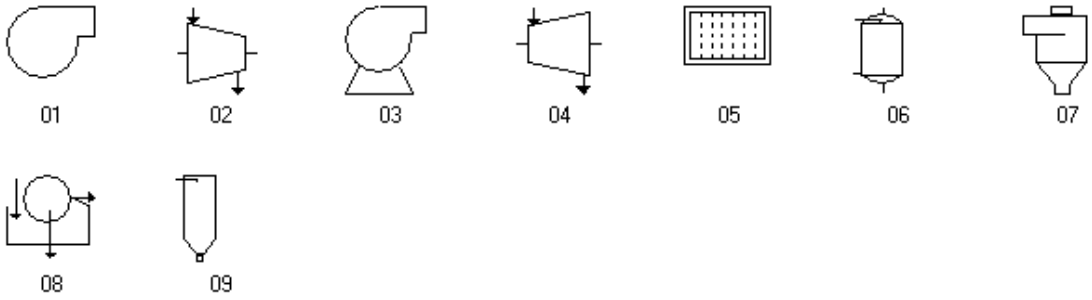
5.5.11.2 isa_s55b



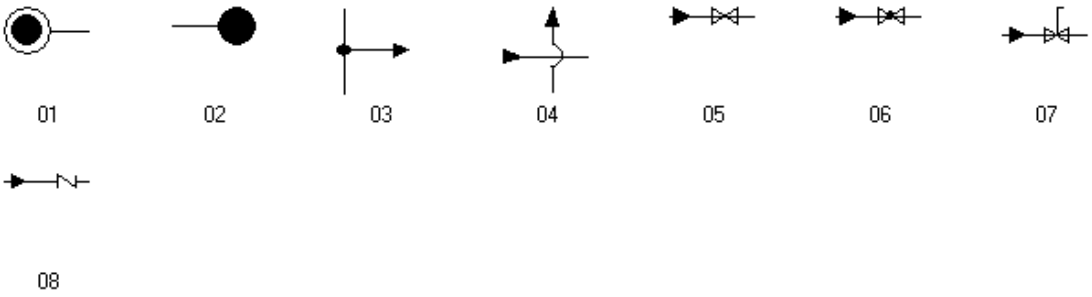
5.5.11.3 isa_s55c



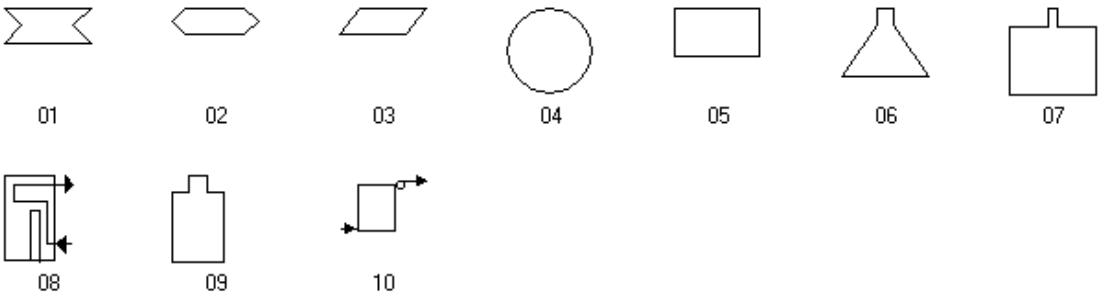
5.5.11.4 isa_s55d



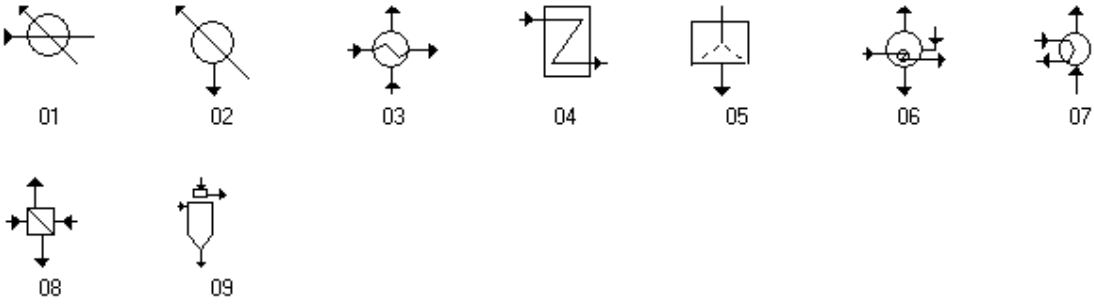
5.5.11.5 isa_y32a



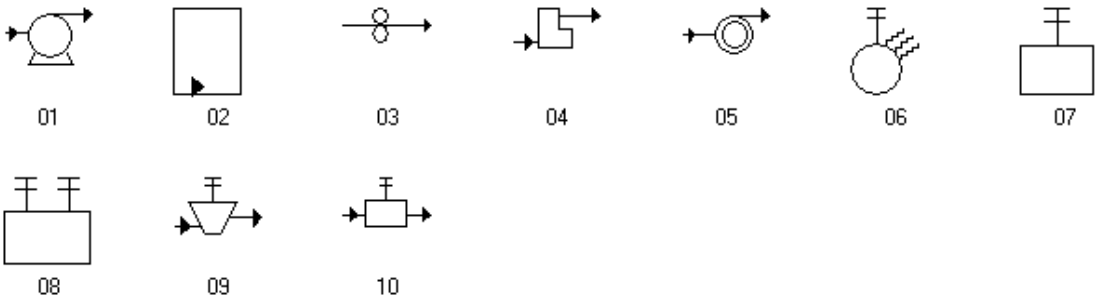
5.5.11.6 isa_y32b



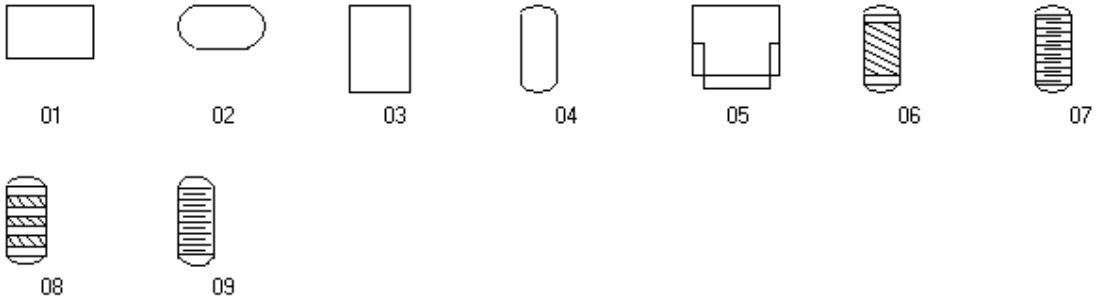
5.5.11.7 isa_y32c



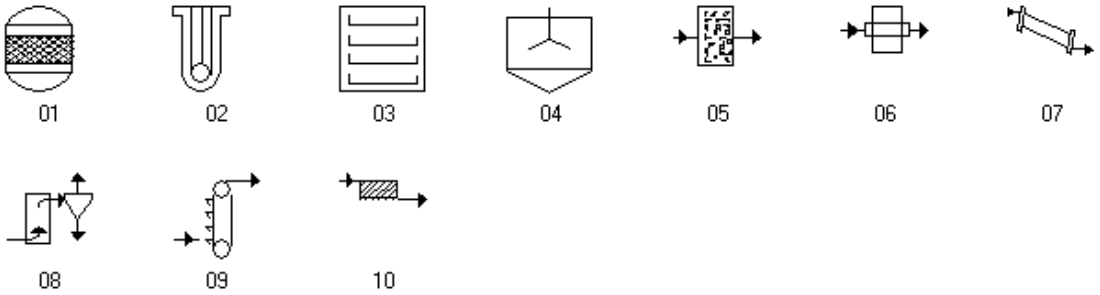
5.5.11.8 isa_y32d



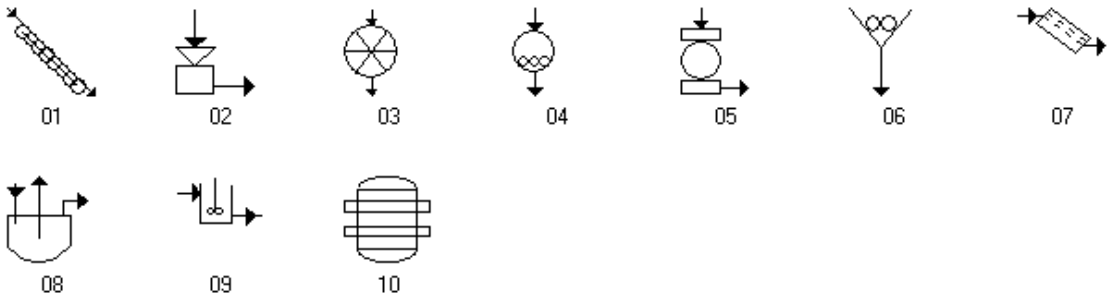
5.5.11.9 isa_y32e



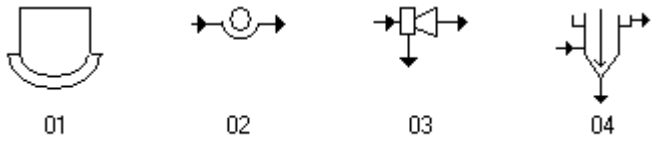
5.5.11.10 isa_y32f



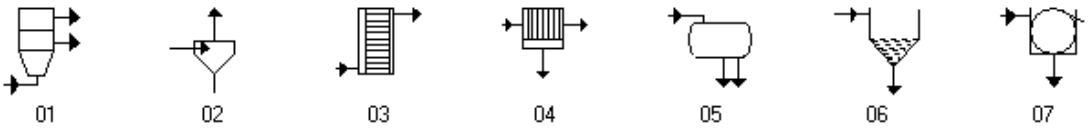
5.5.11.11 isa_y32g



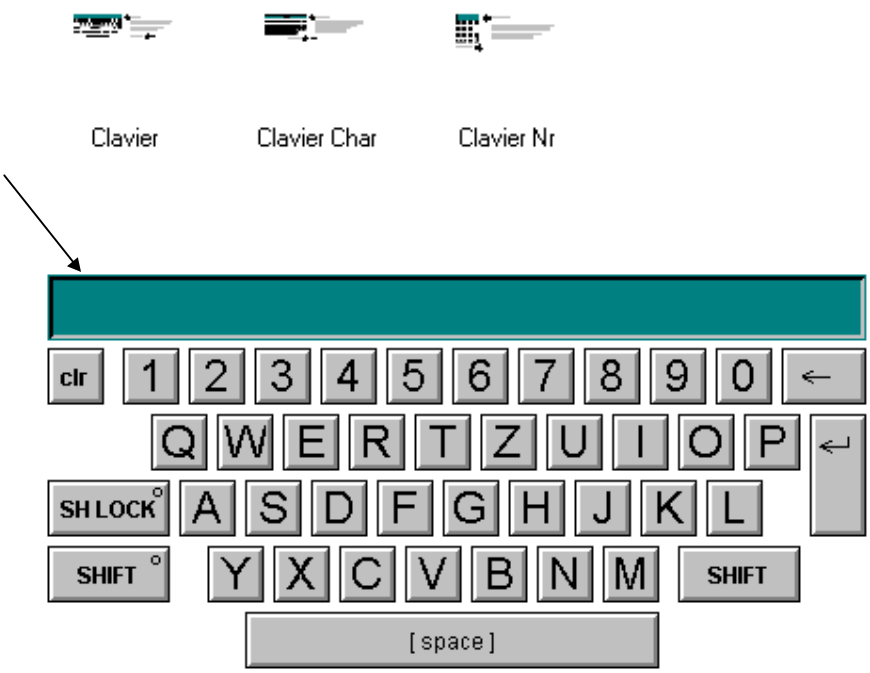
5.5.11.12 isa_y32h



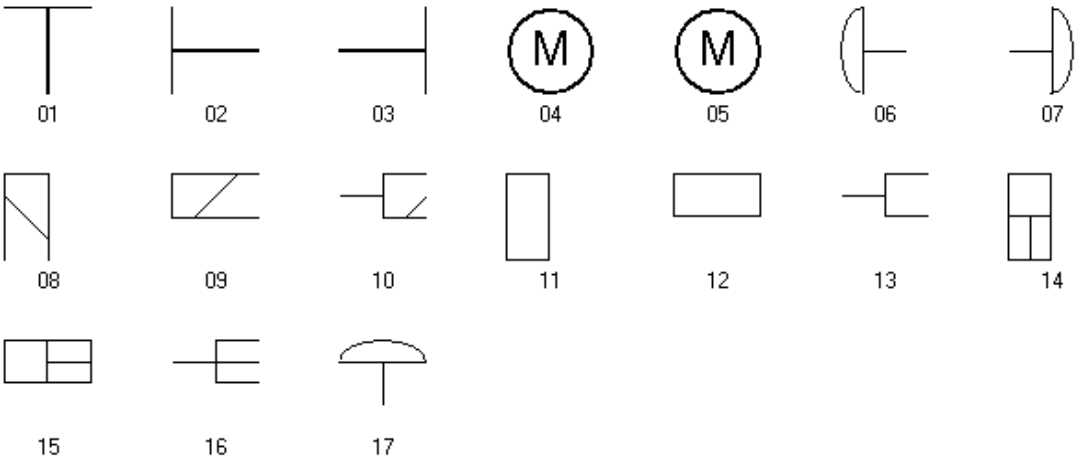
5.5.11.13 isa_y32i



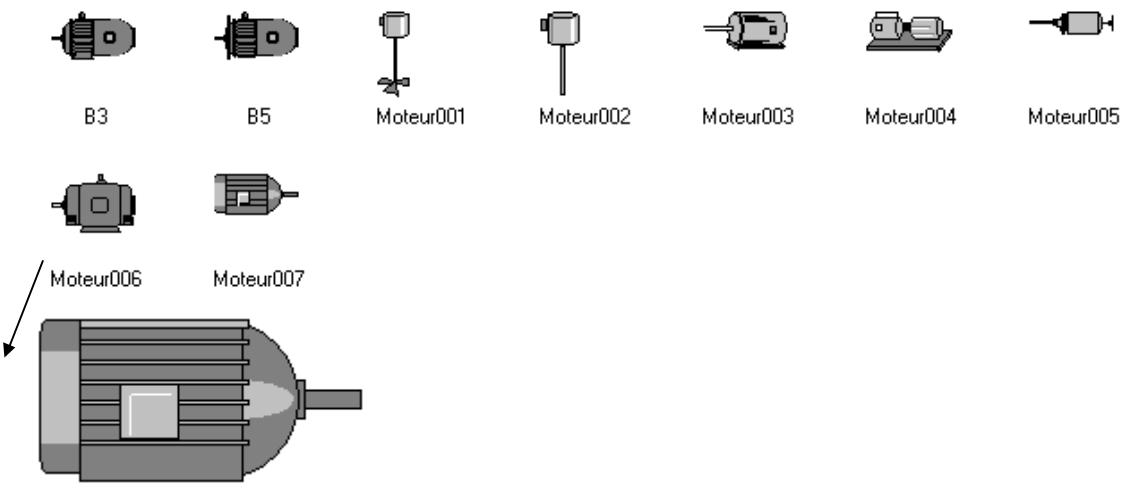
5.5.12 Clavier



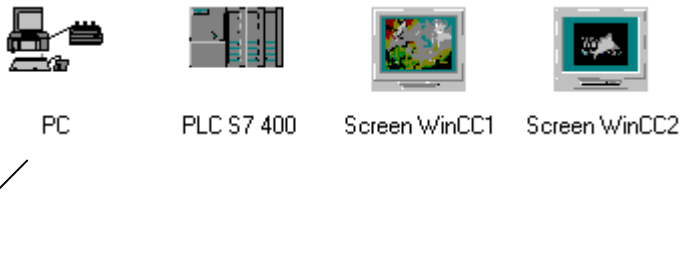
5.5.13 Moteur

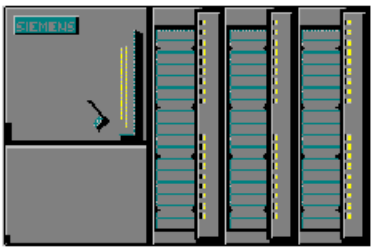


5.5.14 Moteur 3D

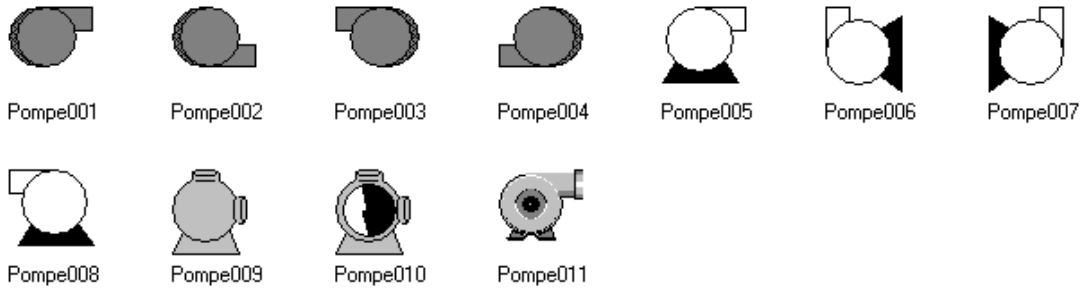


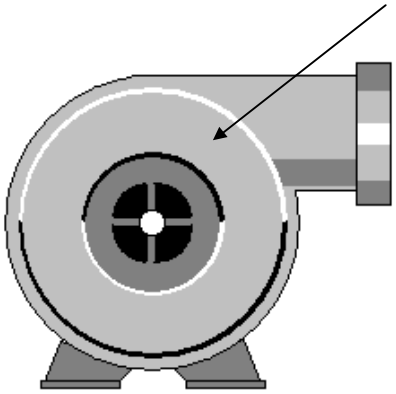
5.5.15 PC / PLC





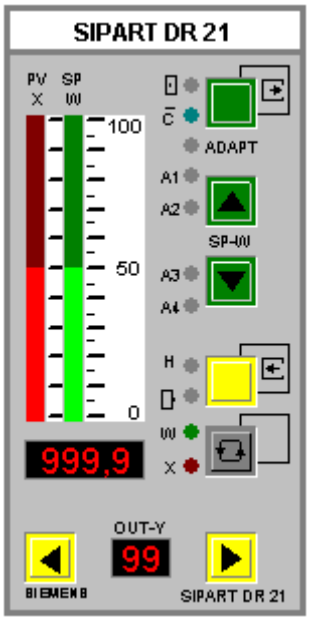
5.5.16 Pompe


Pompe001
Pompe002
Pompe003
Pompe004
Pompe005
Pompe006
Pompe007
Pompe008
Pompe009
Pompe010
Pompe011

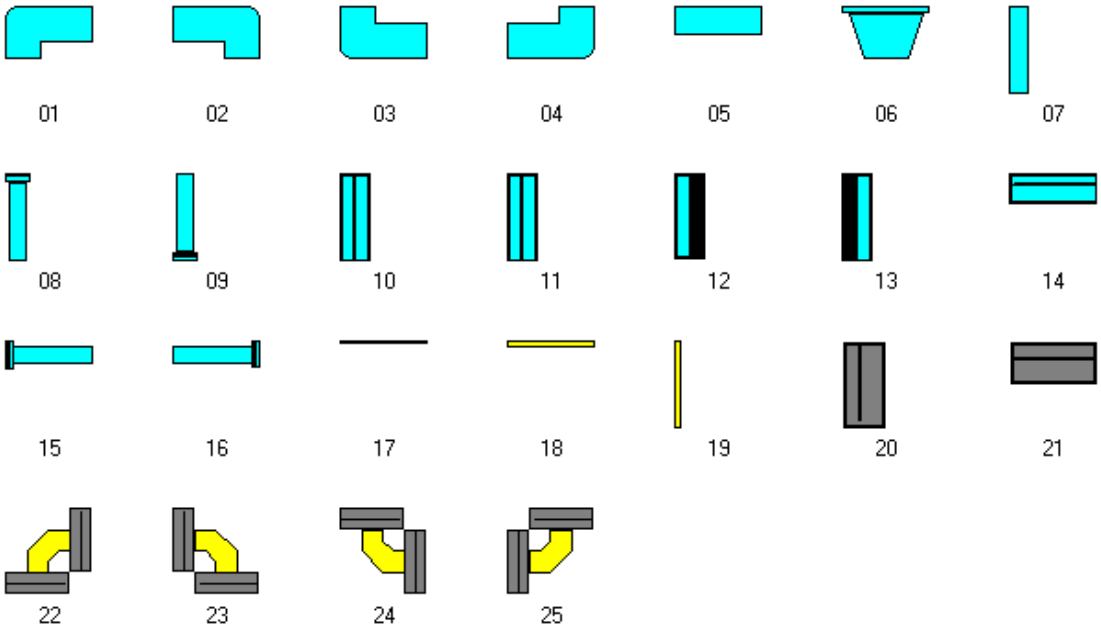


5.5.17 Régulateur

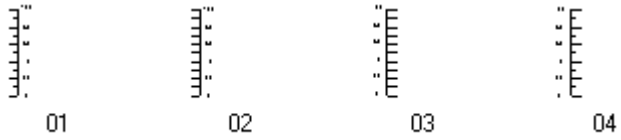
DR 21



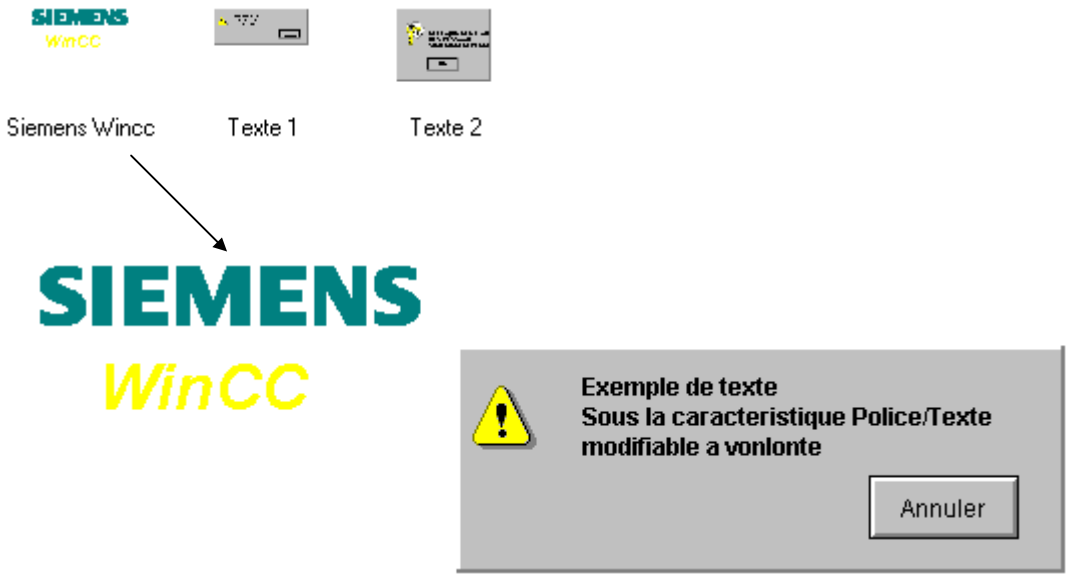
5.5.18 Tuyau


01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

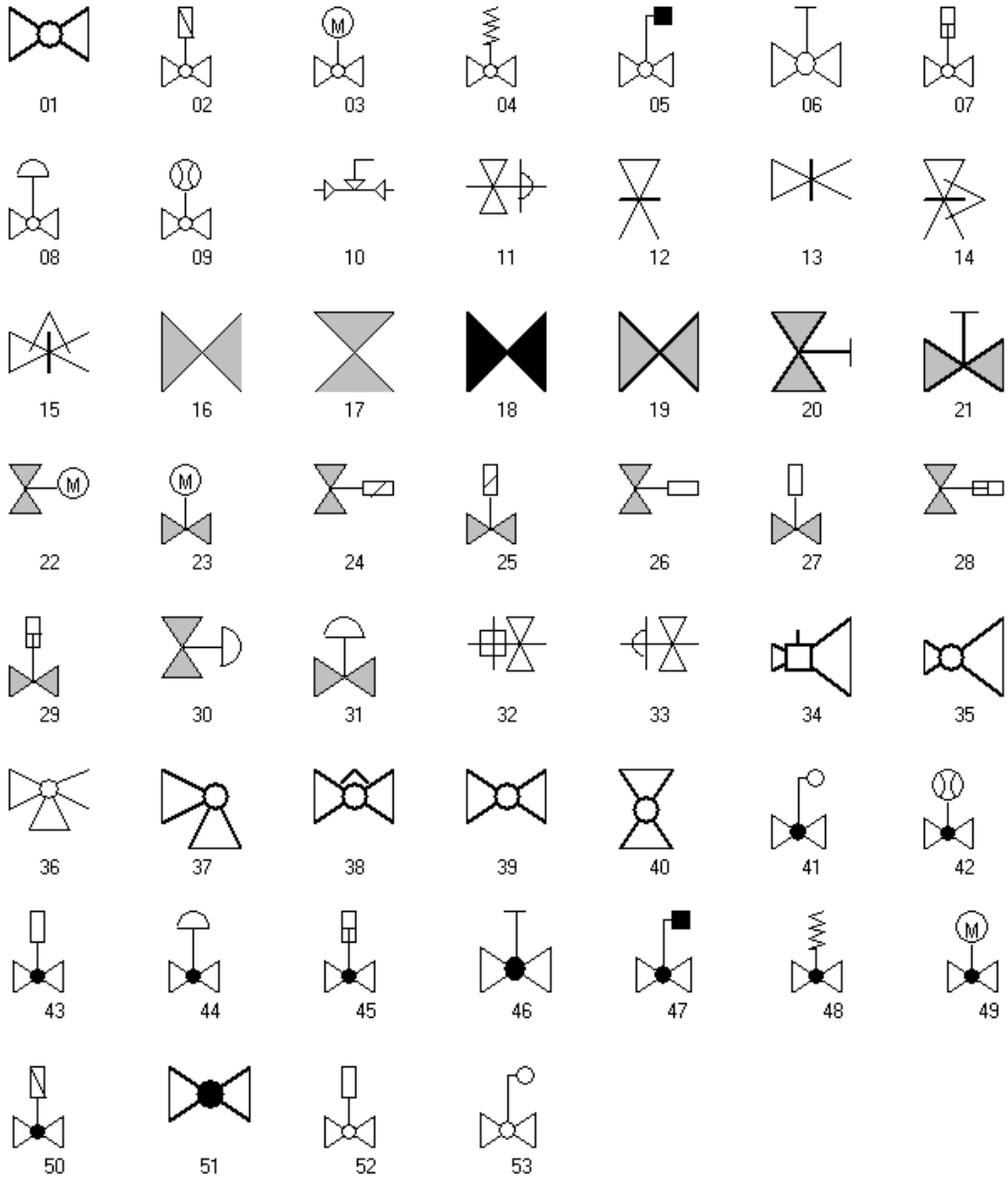
5.5.19 Mise à l'echelle



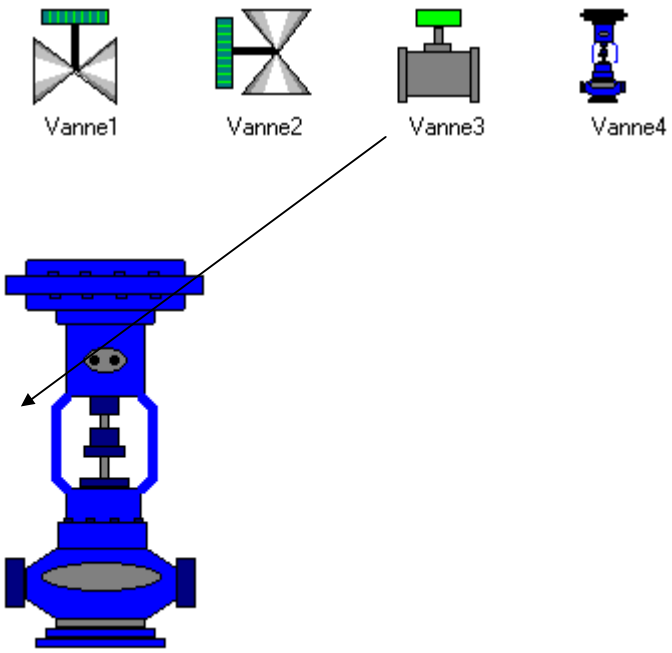
5.5.20 Champ de Texte



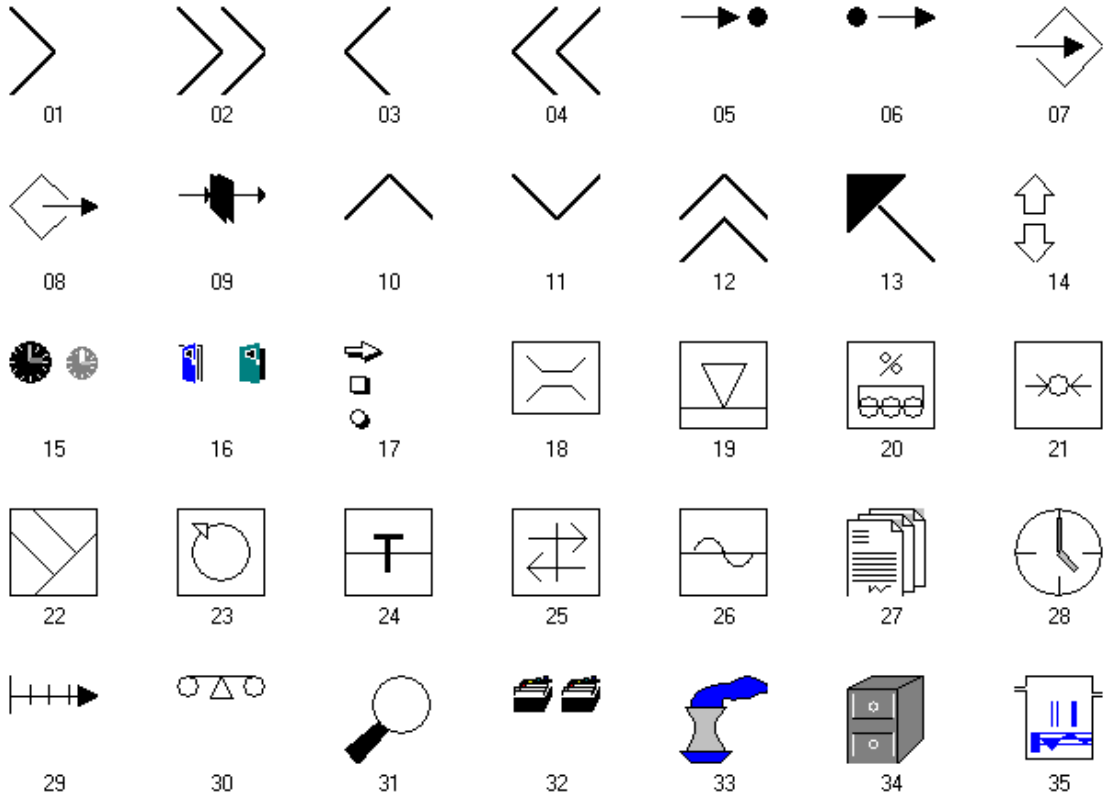
5.5.21 Vanne


01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53

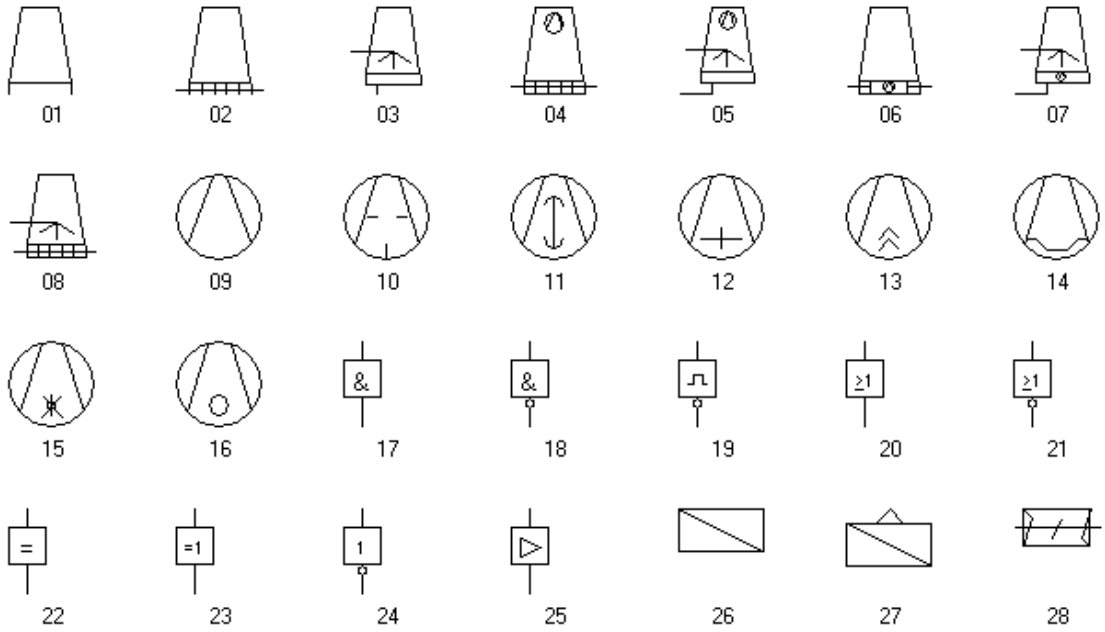
5.5.22 Vanne 3D



5.5.23 Divers1



5.5.24 Divers2



Index

A

Abbréviations, 4-39
Accès
  base de données, 3-46
  Base de données, 3-58
  Droits, 3-52
  Fonctions de protection d'accès, 3-44
Access
  Lire données WinCC, 5-8
Accolade, 4-5, 4-62
Accolades, 4-37
Acquittement
  Alarmes, 3-77
Actions, 4-6
  A la sélection de vue, 5-3
  Conventions, 3-8
  Création, 4-9
  Créer, 4-3
  Cycles de rafraîchissement, 3-24
  dans WinCC, 4-4
  Editeur d', 4-7
  Exportation, 4-12
  Importation, 4-12
  Modifier, 4-6
  Rafraîchissement dans vue, 3-21
  Répertoire, 3-41
  Réutilisation, 3-57
  Réutiliser, 4-6
  Sélection, 3-28
  Tester, 4-11
ActiveX, 2-2, 2-3, 3-48, 3-95
Adaptation
  Données pour importation de variables, 3-62
  Rafraîchissement de vue, 3-21
Adapter
  Propriétés - Ordinateurs, 3-49
Addition, 4-21
Adressage
  indirect, 3-88
  Indirect, 3-5
Affichage
  Code source, 4-72
  Dans la fenêtre de diagnostic, 4-2
Alarme
  Classe d'alarmes, 3-15, 3-65
  Dans concept de conduite, 3-13
  Dynamiser, 4-6
  Ecrire, 3-65
  Liste d'alarmes, 3-77
  Vue d'alarmes, 3-77
Alarmes
  Alarmes chronologiques, 3-18
  Alarmes déclenchées par changement d'état d'un bit, 3-18
  Définition des couleurs, 3-15
  Fichier d'alarmes, 3-65
  Généralités sur la définition, 3-18
  Procédures d'alarme, 3-18
  Réutilisation, 3-63, 3-64
  Sauvegarde, 3-46
  Système d'alarmes, 3-15
Alarmes déclenchées par changement d'état d'un bit, 3-18
Algèbre booléenne, 4-23
ANSI, 4-3
API, 2-4, 3-58, 3-59, 3-96
Appel
  Fonctions, 4-5
Application, 3-21
  Intégration d'applications tierces, 2-3
  Interface (API), 3-58
  Propre, 2-2
  Quitter, 3-43
Archivage, 1-1, 2-2, 3-2, 3-46
  Configuration, 3-50
  Temps d'archivage, 3-21
Archive utilisateur, 3-52, 3-96
  Dynamiser, 4-6
  Réutilisation, 3-67
Arithmétique, 4-21
Array, 4-31
Arrêt, 3-45
ASCII, 4-14
Assistant
  Dans objets utilisateur, 3-89
  Rafraîchissement de vue, 3-21
  Valider Script, 3-96
Assistant Dynamic-Wizard, 3-26
Autorisation, 3-48

B

Barre d'outils
  Alarm Logging, 3-77, 3-80
  Tag Logging, 3-80
Base
  Pour importation, 3-60
  Projet de base, 3-50
  Projet type, 3-67
Base de données, 2-3, 2-4
  Accès avec Access, 5-8
  Accès avec Excel, 5-5
  Accès avec ISQL, 5-9
  Accès avec Scope, 5-10
  Langage d'interrogation, 2-4
  Modification, 3-50
  Restauration, 3-39
  Sauvegarde, 3-46
  Sélection dans, 5-12
Basic, 2-2, 3-95

Basic Process Control, 3-14, 3-73, 3-80
Bibliothèque
Projet, 3-47, 3-54, 3-56
Standard, 3-38
Texte, 3-63
Bibliothèque utilisateur, 3-73
Bitmap
Réutilisation, 3-53
Boucles, 4-37
Do while, 4-37
For, 4-38
While, 4-37
Boucles sans fin, 4-38
Boutons
Configurés, 3-63
Dans fenêtre de courbes, 3-80
Zone de boutons, 3-10, 3-15
Bus système, 3-16

C

Canal
DLL, 3-48, 3-59, 3-62
S7 PMC, 3-97
C-API, 2-4
Caractère spécial, 4-15
Caractères
Admissbles dans noms de vue, 3-7
Admissibles dans nom de variable, 3-6
Caractères autorisés dans les noms de projet, 3-4
Chaîne, 4-5, 4-62
Chaîne de, 4-31
Séquence, 4-62
Caractères spéciaux, 3-7
Cases d'option
Commande par clavier, 3-76
Chaîne de caractères, 4-31
Chaîne de caractètres, 4-36
ChnStart, 5-60
Chronologique
Alarme, 3-18
Clavier
Commande, 3-13, 3-33, 3-69
Touche de raccourci, 3-83
Touches de fonction, 3-72
Clic, 1-3
Client, 2-4
Code source
Affichage, 4-72
Commande, 3-13
Clavier, 3-33, 3-73
Fenêtre de courbes, 3-80
Par touches de fonction, 3-70
Sans souris, 3-69
Séquences des commandes, 1-3
Communication, 3-16, 3-52
Entre tâches, 3-25
Influence sur le rafraîchissement, 3-24
Interface, 3-48
Modification en ligne, 3-97
Process, 3-21
Composants
Intégrer, 2-2
Préconfigurés, 3-55
Technique des composants de vue, 3-85
Composition
Structure C, 4-62
Concept, 2-4
Conduite, 3-10, 3-13
Sauvegarde des données, 3-45
Vues prototypes, 3-86
Concept de conduite, 3-13
Condition
dans les boucles, 4-37
Conditionnelle
Instruction Case, 4-48
Instruction If, 4-46
Conditionnelles
Instructions, 4-37
Conditions, 1-2
Conduite, 3-10
Erreur de conduite, 3-9
Evénementielle, 3-16
Configuration
Conventions pour, 3-2
Dialogue, 3-21, 3-26, 3-35, 3-48, 3-53, 3-89
Données de, 3-40
Dynamisation, 3-35
Mode, 3-42
Connexion, 3-84
Au process, 3-16
Consignes
Pour réutilisation de données, 3-62
Construction, 3-92
Construction de vue, 3-85
Construction image, 3-21
Gestion des variables, 3-5
Structuration des répertoires, 3-19
Contenu, 1-2
Corps de fonction, 4-5
Pointeur, 4-30
Répertoires projet, 3-40
Contraintes, 3-9
Convention, 1-3
Conversion
Monoposte - Multiposte, 3-50
Coros, 3-64
Couleur
Changement, 4-70
Couleurs dans projet, 3-10
Définition, 3-15
Perception, 3-10
Couplage
Série avec CP525, 5-13
Courbes
Commande de fenêtre, 3-77, 3-80

Réutilisation cadre, 3-52
Création
Variable structurée, 4-66
Créer
Fichier avec script, 4-53
Structure d'alarme, 3-66
Curseur, 3-13, 3-75
Curseur, 3-13, 3-75
Pointeur, 3-13, 3-75
Cycle, 3-21
Cycle utilisateur, 3-23, 3-30

D

DDE, 2-2
Déclaration, 4-9, 4-62
Déclencheur, 4-6
Commande temporelle, 3-30
Dans actions C, 3-28
Dans dialogue de dynamisation, 3-28
Rafraîchissement de vue, 3-21
Variables, 3-21, 3-25
Déclencheur temporel, 3-29
Déconnexion, 3-84
Décrémentation, 4-21
Définition
Structures C, 4-61
Syntaxe des pointeurs, 4-30
Démarrage
Entrée dans dossier, 3-42
Messages système pour, 3-37
Vue, 4-1
Déroulement de projets, 1-2, 3-1
Désactiver
Curseur runtime, 3-75
Développement, 2-2
Diagnostic, 4-2, 4-11
Fichier pour importation, 3-60
Fichiers, 3-37
Scope, 5-4
Dialogue
Configuration, 3-21, 3-35, 3-53
Dialogue de dynamisation, 3-22, 3-25
Dialogue dynamisation, 3-34
Division, 4-21
DLL, 4-1, 4-6
Documentation, 1-1, 1-2, 3-2
Domaine
Période, 3-80
Zone de boutons, 3-15
Données
Dans base de données, 3-39
Données, 2-4
Données de configuration, 2-3
Enregistrement, 3-19
Gestion des données, 2-3
Importation, 3-46
Rafraîchissement, 3-16
Ranger, 3-4
Requête, 3-21
Réutilisation, 3-50
Réutilisation S5 ou S7, 3-58
Sauvegarde, 3-47
Séparation, 3-40
Type de données, 4-30
Données de configuration, 2-3
Données de process, 2-4, 3-21
Archivage, 2-2
Droit d'accès, 3-17
Dynamic-Wizard, 3-21, 3-34, 3-85, 3-89
Dynamique
Instance, 3-90
Liaisons aux variables, 3-88
Dynamisation
Configuration, 3-35
Dans WinCC, 3-33
Evénements, 3-34
Mode, 3-25
Modes, 3-21
Modification des propriétés d'objet, 3-33
Objet utilisateur, 3-89
Objets, 3-34
Propriétés, 3-33

E

Ecrire
Alarmes, 3-65
Editeur, 4-7
Editor, 4-5
Einlesen
S5 7 S7 Variablen, 3-58
Elaboration
Concept de conduite, 3-14
Standard d'entreprise, 3-2
EN 60073, 3-15
En ligne
Configuration, 3-96
Enregistrement
OCX, 3-39, 3-95
OLE, OCX, 3-48
En-tête
Générer, 4-9
Entier, 4-15
Entrée
Normalisée, 5-2
Environnement de développement, 3-41, 3-95, 4-3
Erreur
Avertissement, 4-11
de syntaxe, 4-11
Erreur de manipulation, 3-40
Message d'erreur, 3-49
Recherche, 3-37, 4-6
Etablissement
Etablissement de liaison, 3-97
Liaisons, 3-93

Etat
Changement, 3-18
En quittant, 3-42
Indicateur d'état, 3-53
Visualisations d'états, 3-10
Etiquette, 3-12
Evénement
Commande par, 3-16
Déclencheur, 3-21, 3-29, 3-31
Dynamisation, 3-34
En-tête de fonction pour, 4-4
Excel, 3-6, 3-19, 3-60, 3-62, 3-64, 3-66
Lire données avec MSQuery, 5-5
Exécuter
importation-exportation variables, 3-60
Exécution
Scripts, 3-29
Exemple de projet, 1-1, 3-50
Cours de C, 4-2
Langue, 3-3
Résolution d'écran, 3-12
Exigences, 1-1
exportation
Alarmes, 3-39
Exportation, 3-6, 3-39, 3-46, 3-55, 3-60
Actions, 4-12
Fonctions, 4-12
Variables, 3-59
Exportation de données
Avec action C, 5-10
Extension
Actions, 3-57
Fichiers WinCC, 3-40

F

Fclose, 4-53
Fenêtre
Basculement de fenêtre, 3-77
Cycle, 3-23, 3-24, 3-29
Diagnostic, 4-2, 4-11
Fenêtre d'erreur, 4-11
Fenêtre d'alarme, 3-52, 3-77
Fenêtre d'alarmes, 3-52, 3-80
Fenêtre d'application, 2-3, 3-52, 3-77, 3-80
Fenêtre de tableaux, 3-52, 3-80
Fgets, 4-59
Fichiers
Dans répertoire projet, 3-40
Dans répertoire standard, 3-37
Opérations sur fichiers en C, 4-52
FILE, 4-52
Float, 4-14
Fonction Callback
Pour passage de valeurs de process lues, 5-62
Pour transmission en retour de l'état d'un contrat d'écriture, 5-62
Fonctionnalité
Barre d'outils Alarm Logging, 3-80
Barre d'outils Tag Logging, 3-82
Composant de vue, 3-91
Fonctionnalités, 1-1
Fonctions
API, 3-58
Conduite, 3-17
Corps, 4-5
Création, 4-9
dans WinCC, 4-4
Editeur, 4-7
En-tête, 4-4
Exportation, 4-12
Importation, 4-12
internes, 3-24
Internes, 4-6
Projet, 3-41, 4-5
Protection d'accès, 3-44
Réutilisation de fonctions de projet, 3-68
Standard, 3-37, 3-68, 4-5
Structure, 4-4
Tester, 4-11
Fprintf, 4-53
Fscanf, 4-53

G

Générer
Nouvelle en-tête, 3-53, 3-68, 3-93
Recréer en-tête, 3-49
Graphique
Bibliothèque, 3-87
Graphismes
Tools, 3-39
Groupe
Groupe de variables, 3-5
Groupe d'utilisateurs, 3-17, 3-68
Groupes utilisateurs, 3-53
Groupe cible, 4-1

H

Hiérarchie
Conduite, 3-13
Process, 3-70
Vue, 3-73

I

IF, 4-38, 4-46
IHM, 1-1, 2-2, 2-5, 3-16, 3-42
Image
Construction, 3-21
Importation, 3-6, 3-46, 3-58, 3-60
Actions, 4-12
Fonctions, 4-12

Incrémentation, 4-21
Information
Dans vue, 3-9
Masquage, 3-10
Recherche, 3-4
Informix, 2-4
Ingres, 2-4
Initialisation d'une unité de canal
Paramètres pour le gestionnaire de données WinCC, 5-62
Installation
Onduleur, 3-45
Tools, 3-39
WinCC, 3-48
WinCC chemin d'installation, 3-37
Instance
Créer, 3-90
Instanciation, 3-90
Instruction
Conditionnelle, 4-46
Instructions
conditionnelles, 4-37
Instrument à index, 3-10, 3-89
Interface
API, 3-58
Interface de programmation, 2-4
Interface utilisateur, 2-3, 3-2
Conventions, 3-9
Internes
Fonctions, 3-24, 3-37
Intervalle
Intervalle de valeur, 4-14
Introduction
Clavier, 3-69
dans champ d'E/S, 3-75
Moyens, 3-13

L

Lancement
Automatique, 3-42
WinCC automatique, 3-42
Langage
C, C++, 4-30
Langue, 3-63
Réutilisation, 3-63
Lecteur
De sauvegarde de données, 3-46
Liaison
Aux variables de process, 3-93
Etablissement, 3-97
Indirecte, 3-85
Liste de liaisons, 3-62
Logique, 3-58
Nouvelle, 3-85
Liaison directe, 3-21, 3-34, 3-73, 3-87, 3-92
Liaisons
Aux variables, 3-88
Lier
De OCX, 3-85
Liste de démarrage, 4-10
Listes de correspondance, 3-58
Log
Fichiers, 3-37, 3-41
Login, 3-97
Logique, 4-21
Comparaison, 4-21
Logoff, 3-97

M

Manipulable
Fenêtre d'alarme, 3-77
Masquage
Informations, 3-10
Mémoire, 4-31, 4-62
Messages, 2-3
Erreur, 4-11
Messages système, 3-37
Modèle
Fenétre d'alarmes, 3-77
Pour projets utilisateur, 3-3
Vues, 3-91
Modification
Cycle utilisateur, 3-23
Déclencheur, 3-28
Déclencheur sur variable, 3-25
Modifier
Cycle de fenêtre, 3-29
Cycle de vue, 3-28
Etats, 3-33
Fonctions de projet, 4-5
Fonctions standard, 4-5
Propriétés projet, 3-49
Modularité, 2-2
Monoposte
Système, 3-12, 3-45, 3-50
Mot de passe, 3-17, 3-44, 4-6
Multiplication, 4-21
Multiposte
Système, 3-12, 3-45, 3-48, 3-50

N

Nombre, 4-5
à virgule flottante, 4-14
Décimal, 4-15
entiers, 4-14
Format des nombres, 4-11
Nombres à virgule flottante, 4-14

O

Objet utilisateur, 3-89, 3-90, 3-93, 3-94

Assistant, 3-89
Dans technique des composants de vue, 3-85
Réutilisation, 3-54
Objets de commande, 3-75
Objets manipulables
Fenêtre de courbes, 3-80
Objets de commande, 3-75
OCX, 2-3, 3-48, 3-76, 3-95
Composant de vue, 3-85
Enregistrement, 3-39, 3-95
ODBC, 2-2
OLE, 2-2
Liaisons, 3-48
Onduleur, 3-45
Opérateurs, 4-21
Opérations arithmétiques de base, 4-25
Opérations binaires, 4-28
Options, 3-14, 3-39, 3-40, 3-80
Oracle, 2-4
Ordinateur
Modifier type, 3-96
Paramétrage, 3-43
Répertoire, 3-41
Touche de raccourci, 3-77
Outils, 3-73
base de données, 3-46
Importation-exportation de variables, 3-59
Langues, 3-63
OCX, 3-48

P

Par défaut
Langue, 3-38
Répertoire, 3-4
Paramétrage par défaut
Paramétrage, 3-3
Paramètre
Coupure, 3-45
Diagnostic, 3-97
Par défaut, 3-41
Pour printf, 4-15
Paramètres
Liaison, 3-59
Pour nom de variable, 3-6
Pour nom de vue, 3-7
Pour noms de projet, 3-4
Résolution d'écran, 3-12
Performance, 3-50, 4-1
Performances, 1-1, 3-2, 3-16, 3-34
Plage
Plage de valeurs, 3-34
Plate-forme, 2-2
Pointeur, 3-13, 3-75, 4-30
en C, 4-30
Police
Couleur, 3-15
Taille, 3-2, 3-12
Type, 3-2
Printf, 4-2, 4-11, 4-15
Process
Communication, 3-21
Conduite, 3-10
Connexion, 3-16
Hiérarchie de conduite, 3-13
Variables, 3-88
Programme
De réutilisation de données, 3-59
Démarrage, 3-42
Pour base de données, 3-39
Programmes complémentaires, 3-48
spécifique, 3-58
Tools, 3-6
Projet
Bibliothèque, 3-56, 4-12
Conseils pour la réalisation, 3-19
Copier, 3-48
Création d'une fonction, 4-9
Environnement, 3-40
Exemple de projet, 3-3, 4-1
Facilité de maintenance, 3-24
Fonctions, 4-5
Fonctions générales, 3-38
Fonctions utilisables dans tous les, 4-5
Lancement automatique, 3-42
Nom, 3-4
Répertoire, 3-40, 4-10
Réutilisation d'une bibliothèque, 3-54
Sauvegarde, 3-46
Propriété
Dynamisation sur objet, 3-24
En-tête de fonction pour une, 4-4
Objet, 3-22
Propriétés d'une unité de canal
Accès à variable distante, 5-63, 5-70
Ecriture dans adresses de bits, 5-63, 5-70
Ecriture dans adresses d'octets, 5-63, 5-70
Edition des propriétés de canal, 5-63, 5-70
Enregistrement en ligne de liaisons logiques, 5-63, 5-70
Enregistrement en ligne de variables WinCC, 5-63, 5-70
Fonctionnalités client, 5-63, 5-70
Gestion propre de cycle, 5-63, 5-70
Horloge esclave, 5-63, 5-70
Horloge maître, 5-63, 5-70
Pas d'enregistrement de variables WinCC, 5-63, 5-70
Possibilités de diagnostic, 5-63, 5-70
Réentrant, 5-63, 5-70
Signalisation de redémarrage propre, 5-63, 5-70
Surveillance de signe de vie propre, 5-63, 5-70
Valeurs de process décrites avec ordre d'octets INTEL, 5-63, 5-70
Protection d'accès, 3-13, 3-44
Protection des accès
Réutilisation de droits, 3-68

Prototype, 3-85, 3-86, 3-89, 3-91

Q

Quitter
Saisie, 3-76
WinCC, 3-42, 3-45

R

Raccordement
Onduleur, 3-45
Rafraîchissement
Cycles, 3-16
Possibilités de paramétrage, 3-21
Types, 3-24
Recommandation
Pour cycles de rafraîchissement, 3-25
Recopie d'écran, 3-82
Référence
Dans composant de vue, 3-87
Dans composants de vue, 3-94
Dans vues, 3-52
Référence texte, 3-64
Répertoire
Bibliothèque de projets, 3-56
Données dans répertoires WinCC, 3-38
Outils pour WinCC, 3-59
Pour Démarrage, 3-42
Répertoire de projet, 3-4
Répertoire de projet de WinCC, 3-40
Structure de WinCC, 3-19, 3-37
Requête
Données, 3-21
Données par gestionnaire de données, 3-24
Réseau, 3-46, 3-48
Résolution
Ecran, 3-12
Résolution d'écran, 3-12
Restauration
Base de données, 3-39
Restriction
Nom de variable, 3-6
Nom de vue, 3-7
Restrictions
Pour configurations en ligne, 3-96
Pour réutilisation de données, 3-52
Résultat
Dynamisation, 3-35
Sortie, 4-15
Runtime, 3-89, 3-94
Commande dans, 3-75
Conventions pour le, 3-2
Curseur, 3-75
Données, 3-40
Dynamisation, 3-33
Dynamisation au, 3-33
Paramétrage pour, 3-43
Restrictions concernant la configuration en ligne, 3-96

S

Sauvegarde, 3-40
Concept de, 3-45
Données WinCC, 3-46
Scope, 5-4
Accès à la base de données WinCC, 5-10
Scripte
Für Wizard, 3-38
Scripts, 3-8
Editeur de, 4-7
Environnement de développement, 4-3
Exécution, 3-29
Langage, 4-1
Syntaxe, 4-3
Sécurité transactionnelle, 2-3
Sélection, 1-3, 3-13, 3-26, 3-27, 3-28, 3-77
Alarmes, 3-18
Déclencheur, 3-21
Des variables, 3-5
Vues pour indicateur d'état, 3-53
SmartTools, 3-6, 3-39, 3-48, 3-59
Solution
Démarche, 1-1
Lancement automatique de projet, 3-42
Proposition de solution, 1-1
Type de solution, 2-2
Sortie
Normalisée, 5-2
Sortie de test, 4-11
Souris
Action pour touche de raccourci, 3-71
Commande sans souris, 3-69
Dynamisation événements, 3-34
Soustraction, 4-21
SQL, 2-3, 2-4
Base de données, 3-43
Programmation, 3-59
Tools, 3-39
Standard
C, 4-3
Chemin d'installation, 3-37
Contenu répertoire standard WinCC, 3-37
Cycles, 3-25
Fonctions, 3-47, 4-5
Paramétrage, 3-21, 3-23
Touches, 3-72
Structure, 1-2
Base de données, 3-50
Fonctions, 4-4
Listes de textes, 3-61
Manuel de configuration, 1-2
Nom de vue, 3-7
Pour création d'alarmes, 3-66

pour l'enregistrement de données, 3-19
Répertoire de projet de WinCC, 3-40
Répertoire système WinCC, 3-37
Variables, 3-5
WinCC, 2-2
Structures, 4-61
Switch, 4-39
Sybase, 2-3, 3-43
Symbole
Exportation, 3-56
Rangement, 3-38
Réutilisation, 3-53
Type de dynamisation, 3-33
Syntaxe, 4-3
Erreur, 4-11
Messages, 4-11
Systeme
Variables, 3-6
Système
Charge, 3-21, 3-25, 3-32
Composants, 2-2
Environnement, 3-37
Lancement automatique, 3-42
Logiciel, 3-48
Messages, 3-37
Modèle, 3-46
Monoposte, 3-12, 3-50
Multiposte, 3-12, 3-48, 3-50
Plate-forme, 2-2
Système d'exploitation, 3-16
Variables, 3-52
Système d'exploitation, 2-2, 3-16, 3-43, 3-45

T

Table de vérité, 4-23
Table des couleurs, 5-14
Tableaux, 4-31, 4-62
Tâche
Changement de tâche, 3-24
Tâches, 1-1
Barre de tâches, 3-43
Equipe de projet, 3-50
Tampon cyclique
Vues, 3-13
Temps de cycle, 3-24, 3-30
Termes, 2-5
Test
Evénement d'objet, 3-34
Tests
Code clavier, 3-74
Text
Textlibrary, 3-97
Texte
Liste de textes, 3-76
Listes, 3-61
Multilingue, 3-63
Perception à l'écran, 3-10
Visualisation, 3-10
Textes
Listes de textes, 3-58
Tools
Graphisme, 3-39
SQL, 3-39
Variables Importation - Exportation, 3-6
WinCC, 3-39
Touche de raccourci, 3-70, 3-71, 3-77, 3-84
Touches
Affectation des touches, 3-76
Combinaison de touches, 3-42
Dans fenêtre d'alarmes, 3-78
Paramétrage touches, 3-75
Touche de raccourci, 3-71
Touches de commande, 3-53
Touches de fonction, 3-69
Touches de fonction, 3-69
Touches de sens, 3-76, 3-78
Tranformation
Dialogue dynamique en action C, 4-1

U

UNIX, 2-4
Utilisateur, 2-3, 3-9
Autorisation, 3-97
Cycles temporels, 3-23
Enregistrements utilisateur, 2-3
Groupes, 3-68
Réutilisation des droits, 3-68
Utilisateurs
Droits, 3-17
Groupes, 3-17

V

Valeur de mesure
Acquisition, 3-50
Archivage, 3-21
Exportation, 3-39
Rafraîchissement, 3-16
Réutiliser, 3-67
Valeur par défaut
Déclencheur, 3-31
Valeur retournée, 4-4, 4-50
Valeurs analogiques
Affichage, 3-10
Horloge, 3-95
Variables
Dans scripts, 3-6
Déclencheur, 3-21, 3-23, 3-25, 3-31
Définir, 3-5
en C, 4-13
Exportation, 3-59
Importation Exportation, 3-46
Incrustation d'informations dans vue, 3-86

Intervalle de valeur, 4-14
Liaison, 3-35
Messages signalant des variables manquantes, 3-37
Nom, 4-14
Noms admissibles, 3-6
Références dans vues, 3-52
Réutilisation, 3-58
Réutilisation S5 / S7, 3-58
Simulation, 3-41
Structure de listes exportées, 3-61
Types, 4-14
Variables d'archive, 3-81
VDE 0199, 3-15
Vérifications de licence, 3-37
Visual
Basic, 2-2, 3-95
C, 4-11
C++, 2-2, 3-95
Visualisation
Station, 3-21
Visualiser, 2-2
Vue
Changement, 3-70
Construction, 3-85
Construction Fenêtre de vue, 3-92
Contenu de fenêtre, 3-53
Cycle, 3-23
Dimensions de, 3-12
Hiérarchie, 3-73
Informations dans vue, 3-9
Modifier cycle, 3-28
Noms, 3-7
Objet, 3-33
Rafraîchissement, 3-21
Réduire le volume, 4-6
Réutilisation de, 3-51
Sélection, 3-13, 3-53, 3-77, 3-80, 3-94
Technique des composants, 3-85
Vue d'alarmes, 3-77
Vue de process
Commande sans souris, 3-69
Dynamisation, 3-34
Vues synoptiques, 3-13

W

While, 4-37
WinCC
Actions, 3-8
Alarmes, 3-18
API, 3-58
Concept de conduite, 3-13
Copier projet, 3-48
Cycle de rafraîchissement, 3-22
Dynamisation, 3-33
Environnement de projet, 3-40
Environnement système, 3-37
Fichiers Log, 3-37
Gestion des variables, 3-5
Interface utilisateur, 3-9
Lancement automatique, 3-42
Nom de projet, 3-4
Outils, 3-48
Paramétrage par défaut, 3-3
Quitter, 3-45
Répertoire Standard, 3-37
Réutilisation, 3-57
Réutilisation de variables, 3-58
Sauvegarde, 3-46
Scripts, 3-8
Structure, 2-2, 2-3
Structure répertoires, 3-40
Tools, 3-39
Version 1.10, 3-66
Windows, 2-2
Windows 95, 3-16
Windows NT, 3-16
Wizard
Ecrire alarmes Coros, 3-65
Fichiers dans répertoire standard, 3-38
Pour dynamisation d'objets, 3-34
S5 / S7 Ecrire variables, 3-58
Wrebuild, 3-39
Wunload, 3-39

Z

Zone
Ecran, 3-10
Zone de travail, 3-12
Zone d'écran
Zone de travail, 3-12