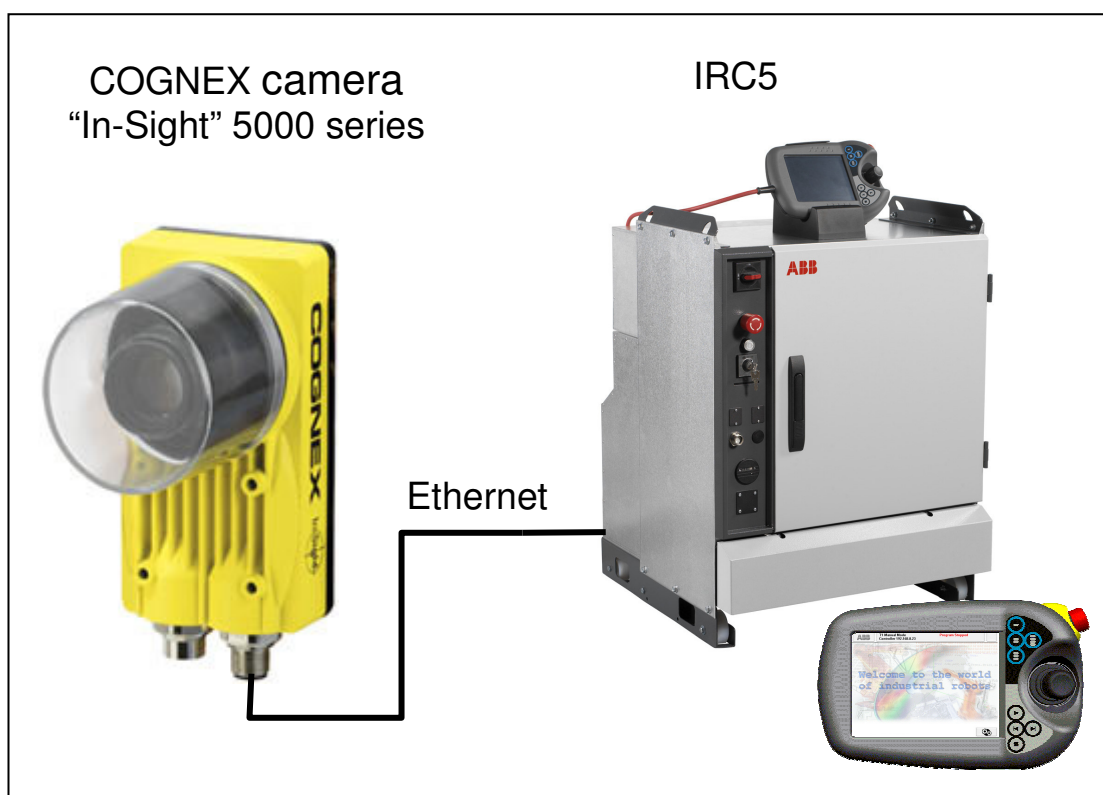




Notice Utilisateur

Interface vision COGNEX « In-Sight » pour système de commande IRC5

IRC5



Notice Utilisateur

Interface vision COGNEX « In-Sight » pour système de commande IRC5

Document ID: RDNU0039_C_InterfaceVisionCOGNEX_FRA
Révision: C – Mise à jour en avril 2009

Les informations contenues dans ce document peuvent être modifiées sans préavis et ne doivent pas être considérées comme un engagement de la part de ABB France s.a.s. ABB France s.a.s réfute toute responsabilité quant aux erreurs éventuelles contenues dans ce document.

En aucun cas, ABB France s.a.s ne peut être tenu responsable des dommages pouvant découler de l'utilisation de ce document ou des logiciels ou matériels décrits dans ce document.

Ce document ne doit pas être reproduit ni copié, intégralement ou en partie, sans l'autorisation de ABB France s.a.s. Il ne doit pas non plus être ni communiqué à des tiers ni utilisé à des fins non autorisées. Toute transgression donnera lieu aux poursuites judiciaires prévues par la loi en vigueur.

D'autres exemplaires de ce document peuvent être obtenus auprès de ABB France s.a.s au prix actuel.

© ABB France 2009

ABB France s.a.s – Division Robotique
Rue de l'Équerre
ZI des Béthunes
95310 Saint-Ouen l'Aumône
FRANCE

Sommaire

Vue d'ensemble	5
Documentation du produit	7
1 Sécurité	8
1.1 A propos du chapitre Sécurité	8
1.2 Standards de sécurité applicables au système de commande IRC5.....	9
1.3 Terminologie sécurité	10
2 Bienvenue	11
2.1 Qu'est ce que l'interface vision COGNEX « In-Sight » pour le système de commande IRC5?	11
2.2 Description des termes et concepts.....	12
3 Organisation du logiciel	14
3.1 Généralités	14
3.2 Chargement du logiciel.....	15
4 Référence des instructions de la librairie de communication	16
4.1 Instructions générales pour une utilisation de base	16
4.1.1 Instruction « CX_SetupCamera »	17
4.1.2 Fonction « CX_InitComm »	19
4.1.3 Fonction « CX_TriggerImage »	20
4.1.4 Fonction « CX_GetVisionData »	21
4.1.5 Instruction « CX_ShowErrStatus »	24
4.1.6 Fonction « CX_CloseComm »	25
4.1.7 Instruction « CX_SetLanguage »	26
4.1.8 Instruction « CX_SetDebugMode »	27
4.1.9 Instruction « CX_ShowVersion »	28
4.2 Instructions de lecture et d'écriture de données dans le programme « In-Sight »	29
4.2.1 Fonction « CX_GetInfo »	30
4.2.2 Fonction « CX_GetValue »	32
4.2.3 Fonction « CX_SetFloat »	34
4.2.4 Fonction « CX_SetInteger »	36
4.2.5 Fonction « CX_SetString »	38
4.3 Instructions d'exécution et de gestion du mode « Online »	40
4.3.1 Fonction « CX_GetOnline »	41
4.3.2 Fonction « CX_SendMessage »	42
4.3.3 Fonction « CX_SetEvent »	44
4.3.4 Fonction « CX_SetEventWait »	46
4.3.5 Fonction « CX_SetOnline »	48
4.4 Instructions de gestion de fichiers	50
4.4.1 Fonction « CX_GetFile »	51
4.4.2 Fonction « CX_LoadFile »	53
4.5 Instructions d'étalonnage robot-vision	54
4.5.1 Instruction « SetToolVision »	55
4.5.2 Instruction « SetWObjVision »	57

5	Mise en œuvre des instructions de la librairie de communication	59
5.1	Séquences type d'utilisation des instructions de la librairie	59
5.2	Utilisation de la transformation de vision	61
5.3	Programmes d'exemple fournis avec la librairie de communication	65
5.4	Configuration de l'adresse IP pour un système de commande IRC5	66
5.5	Configuration de l'adresse IP pour une caméra « In-Sight »	69
6	Interface Homme-Machine "IHM"	70
6.1	Messages d'erreurs générés par la bibliothèque de communication ABB	70
6.2	Messages d'erreurs générés par la caméra « In-Sight »	71
6.3	Message d'information généré par les fonctions de la librairie de communication	72
6.4	Affichage d'image de caméra « In-Sight » sur le pupitre « FlexPendant »	73
7	Procédures d'étalonnage robot – vision	74
7.1	Mise en œuvre d'un système de vision pour le guidage d'un robot	74
7.2	Etalonnage robot – vision dans le cas d'une caméra fixe	75
7.3	Etalonnage robot – vision dans le cas d'une caméra embarquée sur le robot.....	79
8	Procédures particulières	80
8.1	Utilisation du « Robot Virtuel » pour tester une application de vision	80
8.2	Mode de mise au point « debug »	82
8.3	Transfert d'image de la caméra « In-Sight » vers le système de commande IRC5	83
9	Evolutions logicielles	84
10	Index et tables	85
10.1	Index.....	85
10.2	Table des illustrations	85

Vue d'ensemble

A propos de ce manuel

Ce manuel détaille les instructions relatives à la mise en œuvre de l'interface vision COGNEX « In-Sight » pour un système de commande IRC5 équipé du pupitre « FlexPendant ».

Ce manuel fait référence à la version **1.0.3** de l'interface.

Usage

Ce manuel doit être utilisé pendant la programmation ou la maintenance d'une application de vision à base de caméras COGNEX « In-Sight ».

Qui doit lire ce manuel ?

Ce manuel est rédigé pour :

- Les programmeurs robot,
- Les développeurs d'applications robot,
- Les techniciens de service.

Pré requis

Le lecteur doit :

- Etre familiarisé avec le langage RAPID,
- Avoir été formé à l'utilisation du logiciel RobotStudio Online et du pupitre « FlexPendant »,
- Avoir été formé à l'utilisation du logiciel « In-Sight Explorer » de la société COGNEX, ainsi qu'à la mise en œuvre des caméras COGNEX « In-Sight ».

Références

Référence	Document ABB
Manuel Produit, procédures	IRC5 3HAC 021313-001
Manuel Produit, références	IRC5 3HAC 021313-001
Démarrage - IRC5 et RobotStudio Online	3HAC 021564-001
Manuel opérateur - RobotStudio Online	3HAC 18236-1
Manuel de référence RAPID – Vue d'ensemble RAPID	3HAC 16580-1
Manuel de référence RAPID – Instructions	3HAC 16581-1
Manuel de référence RAPID – Fonctions et types de données	3HAC 16581-1
Manuel de référence RAPID – RAPID Kernel	3HAC 16585-1

Reference	Document COGNEX
"In-Sight" Explorer 3.2 Reference Guide	Revision: 3.2.1219
"In-Sight" Explorer 4.1 Reference Guide & EasyBuilder	Revision: 4.1.0

Comment lire ce manuel ?

Afin de faciliter la lecture, le texte ayant une signification particulière adopte un formatage spécifique.

Type de texte	Formatage	Exemple
Eléments de l'interface utilisateur, comme menu, boîte de dialogue, bouton.	Gras	Dans la boîte de dialogue Ouvrir Fichier , cliquez sur OK
Touches du clavier	Capitales	Pour lancer l'aide en ligne, pressez la touche F1.
Messages ordinateurs et instruction RAPID	Monospace (courrier)	MoveJ
Variables dans les messages Instructions RAPID	Monospace (courrier) italique	PROC <i>RoutineName</i> ()
Liens hypertexte	Bleus italiques	Voir le manuel RobotStudio Online Operator's Manual
Références à autres manuels ou sources d'info	Italique	Voir le <i>Manuel de référence RAPID</i> pour plus d'information.

Révisions

Révision	Description	Rédacteur
A	Première édition – mars 2007	Philippe CHARLES
B	Deuxième édition – janvier 2008	Philippe CHARLES
C	Troisième édition – avril 2009	Philippe CHARLES
D		

Documentation du produit

Généralités

La documentation robot est divisée en différentes catégories. Cette liste est basée sur le type d'information contenu dans chaque document, que les produits sont optionnels ou pas. Ceci signifie que chaque livraison de produit *robot ne contient pas tous* les documents listés, mais seulement ceux correspondant à l'équipement livré.

Cependant, tous les documents listés peuvent être commandés auprès d'ABB. Les documents listés sont valables pour les systèmes robot M2004.

Manuels

Les manuels suivants décrivent le logiciel robot en général et contient les informations de références applicables.

- **RAPID Vue d'ensemble** : vue d'ensemble du langage de programmation RAPID.
- **RAPID Manuel de référence** : description de toutes les instructions et fonctions RAPID, ainsi que des types de données.
- **Manuel de référence technique – paramètres systèmes**: description des paramètres systèmes.

1 Sécurité

1.1 A propos du chapitre Sécurité

Introduction à la Sécurité

Ce chapitre décrit les principes de sécurité et les procédures à utiliser quand un robot ou un système robot est en opération.

Il ne couvre ni la conception ni l'installation de la sécurité de l'équipement. Ces thèmes se trouvent dans les Manuels Produit livrés avec le système robot.

Sécurité du personnel

Un robot est lourd, extrêmement puissant et rapide. Une pause ou un arrêt du mouvement peut être suivi d'un mouvement rapide aléatoire. Même si une trajectoire est prévue, une modification en cours d'opération peut intervenir via un signal externe résultant d'un mouvement inattendu. Il est donc important de suivre toutes les règles en vigueur quand on pénètre dans l'espace de sécurité du robot.

Régulation sécurité

Avant de commencer à travailler avec le robot, assurez-vous que vous de connaître les règles de sécurité décrites dans le *Manuel Opérateur – IRC5 avec FlexPendant*.

1.2 Standards de sécurité applicables au système de commande IRC5

Standards de sécurité

Le robot est totalement conforme aux standards de sécurité spécifiés dans les directives européennes machine. Les robots ABB commandés par le système de commande IRC5 sont conformes aux standards suivants :

Standard	Description
EN ISO 12100-1	Safety of machinery, terminology
EN ISO 12100-2	Safety of machinery, technical specifications
EN 954-1	Safety of machinery, safety related parts of control systems
EN 775	Manipulating industrial robots, safety
EN 60204	Electrical equipment of industrial machines
EN 61000-6-4 (option)	EMC, generic emission
EN 61000-6-2	EMC, generic immunity

Standard	Description
IEC 204-1	Electrical equipment of industrial machines
IEC 529	Degrees of protection provided by enclosures

Standard	Description
ISO 10218	Manipulating industrial robots, safety
ISO 9787	Manipulating industrial robots, coordinate systems and motions

Standard	Description
ANSI/RIA 15.06/1999	Safety requirements for industrial robots and robot systems
ANSI/UL 1740-1998 (option)	Safety standard for robots and robot equipment
CAN/CSA Z 434-03 (option)	Industrial robots and robot systems - General safety

1.3 Terminologie sécurité

Vue d'ensemble

Cette section indique tous les types d'avertissement pouvant intervenir lors des opérations décrites dans ce manuel. Chaque avertissement est expliqué dans sa propre section avec :

- Un symbole par niveau d'avertissement (DANGER, AVERTISSEMENT ou PRUDENCE) et type de danger
- Une brève description de ce qui arrive si l'opérateur/personnel de service n'élimine pas le danger
- Une instruction permettant d'éliminer le danger et de réaliser l'opération manuellement.

Niveaux de danger

Le tableau ci-dessous définit les symboles des avertissements utilisés dans ce manuel.

Symbole	Désignation	Signification
	Danger	Avertit des risques qui causeront des blessures personnelles graves, voire mortelles et/ou l'endommagement majeur du produit si les consignes ne sont pas respectées.
	Avvertissement	Avertit des risques pouvant causer des blessures personnelles ou l'endommagement majeur du produit. Toujours respecter les consignes accompagnant ce symbole.
	Choc électrique	Avertit des risques de choc électrique pouvant causer des blessures personnelles graves, voire mortelles. Toujours respecter les consignes accompagnant ce symbole.
	Prudence	Attire l'attention sur le risque d'endommagement du produit si une certaine action n'est pas effectuée ou est effectuée de manière incorrecte.
	Décharges électrostatiques (ESD)	Le symbole ESD indique le risque de décharges électrostatiques pouvant entraîner un endommagement majeur du produit.
	Conseil	Dirige vers des instructions spécifiques pour trouver plus d'information, ou astuces pour réaliser la même opération d'une façon plus simple.
	Remarque	Information sur des éléments importants.

2 Bienvenue

2.1 Qu'est ce que l'interface vision COGNEX « In-Sight » pour le système de commande IRC5?

Description de l'interface vision COGNEX « In-Sight »

L'interface vision se présente sous la forme d'une librairie de routines en langage « RAPID » qui prend en charge les fonctions de communication entre le système de commande IRC5 et une ou plusieurs caméras COGNEX type « In-Sight » de la série « 5000 », connectés par un réseau Ethernet. Une même caméra peut également communiquer avec plusieurs systèmes de commande IRC5.

Principaux éléments

Principaux éléments de l'interface vision COGNEX « In-Sight » :

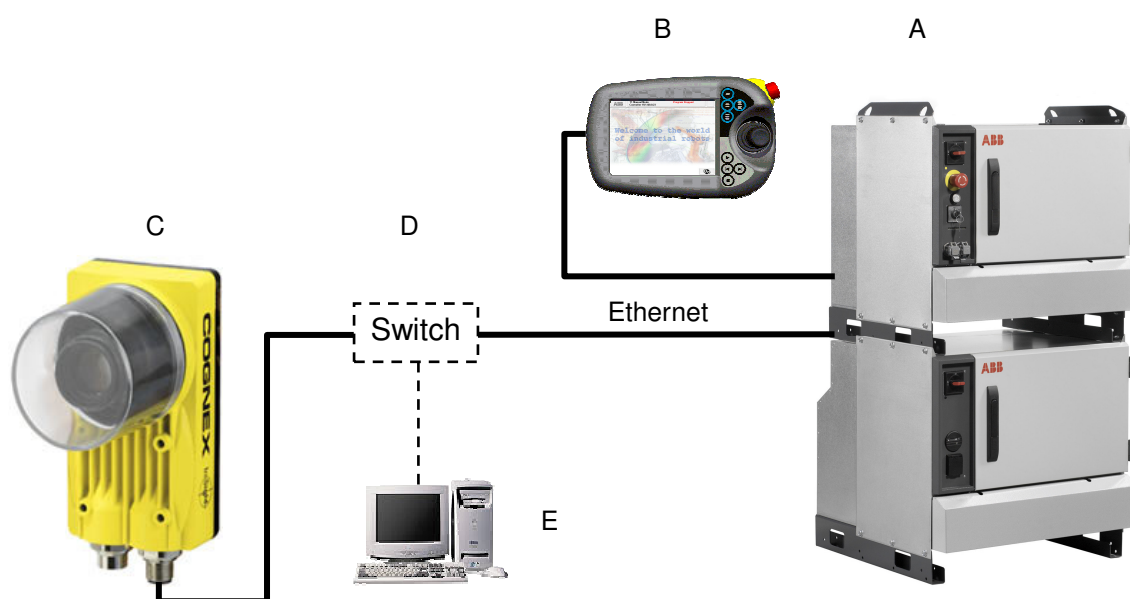


Figure 1. Architecture matérielle avec un système de commande IRC5

A	Système de commande IRC5
B	Pupitre graphique « FlexPendant »
C	Caméra « In-Sight » série « 5000 »
D	Hub ou « switch » Ethernet
E	Ordinateur compatible PC

Note : les éléments (D) et (E) ne sont nécessaires que pendant la mise au point du programme de vision avec le logiciel « In-Sight Explorer » fourni par la société COGNEX.

2.2 Description des termes et concepts

Vue d'ensemble

Ce chapitre contient la description des principaux concepts et mots utilisés dans ce manuel. Il peut également y avoir des informations supplémentaires dans tous les autres chapitres.

Programme robot

Le programme robot d'un système de commande IRC5 est écrit en langage structuré « RAPID ».

Ce programme peut être visualisé et édité sur un ordinateur compatible PC avec le logiciel « RobotStudio Online » ou sur le pupitre « FlexPendant » du robot.

Programme vision

Le programme vision d'une caméra « In-Sight » est appelé « job » et se présente sous la forme d'un tableur avec 400 lignes (de 0 à 399) et 26 colonnes (de A à Z).

Ce programme peut être visualisé et édité sur un ordinateur compatible PC avec le logiciel « In-Sight Explorer » fourni par la société COGNEX.

Vision 2D

La vision « 2D » signifie vision en 2 dimensions : l'image obtenue par la caméra est traitée dans un plan. Les coordonnées obtenues sont 2 translations (X et Y) et une rotation autour de la normale Z du plan image (rotation RZ).

Dans le cas du traitement d'image COGNEX, ces 3 coordonnées sont complétées par un indice de fiabilité de reconnaissance de l'objet, appelé « score » et exprimé sous la forme d'un pourcentage (compris entre 0 et 100%).

Transformation

Une transformation est un objet mathématique qui décrit la combinaison d'une translation et d'une rotation dans l'espace. La transformation est utilisée pour définir la relation géométrique entre deux repères.

La transformation peut être exprimée sous une forme matricielle (matrice « homogène » de dimension 4x4) ou vectorielle. C'est le second formalisme qui est utilisé dans le langage « RAPID ».

« Socket », port de communication et adresse IP

Le « socket » est un mécanisme de communication qui permet l'échange de messages entre 2 processus informatiques, en général situés sur des ordinateurs différents connectés en réseau. Dans ce cas, chaque « socket » utilise un port de communication et une adresse « IP » :

- Le port de communication est un port logique défini par un nombre entier codé sur 16 bits. Les valeurs généralement réservées au système sont comprises entre 0 et 1023, les valeurs supérieures à 1023 sont disponibles pour l'utilisateur, en général pour des connections temporaires.
- L'adresse « IP » est une adresse logique qui permet d'identifier un ordinateur connecté en réseau « IP ». L'adresse « IP » est définie par un nombre entier 32 bits écrit en décimal sous la forme de 4 octets (de 0 à 255) séparés par un point. Exemple : « 10.33.50.105 ».

Un « socket » peut être utilisé selon 2 modes :

- Le mode « connecté » qui utilise le protocole de transport « TCP »,
- Le mode « non connecté » qui utilise le protocole de transport « UDP ».

Le système de commande IRC5 permet d'utiliser les « sockets » selon les 2 modes. L'interface vision COGNEX « In-Sight » pour le système de commande IRC5 n'utilise que le mode « connecté ».

Protocole d'application « Telnet »

Le protocole de communication « Telnet » est un protocole d'application standard d'Internet. Ce protocole à base de chaînes de caractères permet de relier un « client » (ex : un terminal de saisie) et un « serveur » (ex : un ordinateur distant).

Ce protocole de communication est utilisé sur un très grand nombre d'ordinateurs, car il est disponible sur la plupart des systèmes d'exploitations (UNIX, Windows ...) disponibles sur les ordinateurs personnels, ainsi que sur du matériel informatique industriel : c'est le cas des caméras COGNEX « In-Sight ».

Dans le cas du système de commande IRC5, le protocole « Telnet » met en œuvre les « sockets » de communication associé au protocole de transport de données « TCP ».

Le port de communication utilisé par défaut pour le protocole « Telnet » est le numéro 23.

3 Organisation du logiciel

3.1 Généralités

Vue d'ensemble

L'interface vision est fournie sous la forme de 7 modules systèmes écrits en langage RAPID :

- INSIGHT Module général (obligatoire)
- INSIGHT_DUT Module pour la localisation en Hollandais
- INSIGHT_ENG Module pour la localisation en Anglais (obligatoire)
- INSIGHT_FRA Module pour la localisation en Français
- INSIGHT_GER Module pour la localisation en Allemand
- INSIGHT_ITA Module pour la localisation en Italien
- INSIGHT_SPA Module pour la localisation en Espagnol
- INSIGHT_SWE Module pour la localisation en Suédois

Ces modules sont cryptés et ne sont pas éditables par l'utilisateur. Ces modules sont disponibles soit sous forme de « module programme » ou de « module systèmes » : c'est à l'utilisateur de choisir quel type de module est le plus approprié en fonction de l'architecture de son application.

Option logicielle nécessaire pour le système de commande IRC5

Le fonctionnement de l'interface vision COGNEX « In-Sight » nécessite la fonction de communication « Socket messaging » avec l'une des options logicielles suivantes :

- RobotWare 5.06 à 5.09 : « Socket Messaging » référence [672-1] ;
- RobotWare 5.10 et suivant : « PC Interface » référence [616-1] ;

Veuillez vérifier que votre système de commande IRC5 est bien équipé de cette option logicielle.



Conseil :

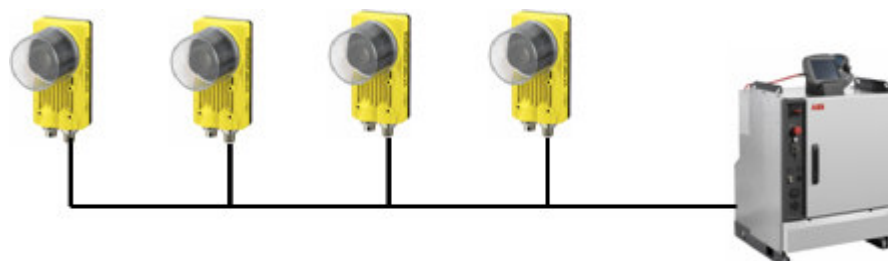
Pour plus de détails sur l'option logicielle « Socket Messaging », veuillez vous référer à :

- La documentation Française « Caractéristique du produit – Logiciel du système de commande IRC5 » référence 3HAC022349-001_rev3_FR, ou à
- La documentation Anglaise « Product Specifications – Controller software IRC5 » référence 3HAC022349-001_rev3_EN.

Architecture « client/serveur »

La communication entre un système de commande IRC5 et les caméras COGNEX « In-Sight » est réalisée par le réseau Ethernet et utilise le protocole de communication « Mode Natif » propre aux caméras « In-Sight ».

La caméra est toujours utilisée comme serveur de communication et le robot comme client. Cela signifie que le robot est toujours à l'initiative de la communication et envoie des requêtes à la caméra, qui ensuite répond à la demande. Dans cette version, le robot peut dialoguer avec 4 caméras.



3.2 Chargement du logiciel

Vue d'ensemble

Le logiciel et la documentation sont livrés sous la forme d'un fichier compressé « ZIP ». Séquence des opérations à réaliser :

- Décompresser le fichier : le produit est structuré sous forme de plusieurs répertoires,
- Décider si vous voulez utiliser des modules. »programmes » ou des modules « systèmes » (recommandé par défaut),
- Décider quelle langue d'interface homme-machine vous allez utiliser en plus de l'Anglais (qui est obligatoire),
- Sélectionner en conséquence les modules dans la librairie de répertoires, par exemple « INSIGHT_FRA » pour le Français. Vous devez dans tous les cas charger les modules « INSIGHT » et « INSIGHT_ENG », pour exécuter l'interface sans erreur.

Les modules nécessaires peuvent être chargés dans la mémoire du système de commande IRC5 de deux façons en utilisant :

- Un ordinateur compatible PC et le logiciel « RobotStudio Online », relié au système de commande IRC5 par la liaison Ethernet sur le port « Service » ou « LAN »,
- Le pupitre de programmation « FlexPendant » qui peut charger les fichiers depuis une clé USB.

4 Référence des instructions de la librairie de communication

4.1 Instructions générales pour une utilisation de base

Vue d'ensemble

Ce chapitre décrit les instructions suivantes:

- CX_CloseComm
- CX_GetVisionData
- CX_InitComm
- CX_SetDebugMode
- CX_SetLanguage
- CX_SetupCamera
- CX_ShowErrStatus
- CX_ShowVersion
- CX_TriggImage

Ces instructions correspondent aux opérations les plus fréquemment utilisées pour gérer la communication entre le robot et une ou plusieurs caméras.

4.1.1 Instruction « CX_SetupCamera »

CX_SetupCamera : configuration des données associées à une caméra

L'instruction "CX_SetupCamera" définit les données de configuration d'une caméra concernant la communication et les tolérances de reconnaissance d'un objet.

Cette instruction doit être utilisée en premier lieu, avant toutes les autres instructions de la librairie de communication. La caméra doit avoir été configurée au préalable avec l'outil de la société COGNEX pour définir son adresse IP.

Exemple

```
CX_SetupCamera 1, "10.33.50.100", 23, "admin", "", -80, 80, -60, 60, -90, 90, 70;
```

Arguments

CX_SetupCamera Identity SocketIP SocketPort UserName Password MinCoordX MaxCoordX
MinCoordY MaxCoordY MinCoordRZ MaxCoordRZ MinScore

Identity

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

SocketIP

Type de donnée : string

Chaîne de caractères qui définit l'adresse IP de la caméra : « xxx.xxx.xxx.xxx » (4 octets compris entre 0 et 255 et séparés par un point).

SocketPort

Type de donnée : num

Valeur entière qui définit le port de communication pour dialoguer avec la caméra.

Note : le port utilisé par défaut pour le protocole « TelNet » est le **23**.

UserName

Type de donnée : string

Chaîne de caractères qui définit un nom d'utilisateur associé à la caméra pendant l'utilisation en production.

Note : le nom d'utilisateur utilisé par défaut sur les caméras « In-Sigh » est « **admin** ». Une caméra peut gérer jusqu'à 16 utilisateurs différents.

Password

Type de donnée : string

Chaîne de caractères qui définit le mot de passe associé à « UserName ».

Note : le mot de passe de l'utilisateur par défaut « admin » sur les caméras « In-Sigh » est une chaîne vide.

MinCoordX

Type de donnée : num

Valeur réelle qui définit le seuil minimum de la coordonnée de translation X d'un objet dans le repère caméra. Une coordonnée inférieure sera considérée comme incorrecte.

MaxCoordX

Type de donnée : num

Valeur réelle qui définit le seuil maximum de la coordonnée de translation X d'un objet dans le repère caméra. Une coordonnée supérieure sera considérée comme incorrecte.

MinCoordY

Type de donnée : num

Valeur réelle qui définit le seuil minimum de la coordonnée de translation Y d'un objet dans le repère caméra. Une coordonnée inférieure sera considérée comme incorrecte.

MaxCoordY

Type de donnée : num

Valeur réelle qui définit le seuil maximum de la coordonnée de translation Y d'un objet dans le repère caméra. Une coordonnée supérieure sera considérée comme incorrecte.

MinCoordRZ

Type de donnée : num

Valeur réelle qui définit le seuil minimum de la coordonnée de rotation RZ d'un objet dans le repère caméra. Une coordonnée inférieure sera considérée comme incorrecte.

MaxCoordRZ

Type de donnée : num

Valeur réelle qui définit le seuil maximum de la coordonnée de rotation RZ d'un objet dans le repère caméra. Une coordonnée supérieure sera considérée comme incorrecte.

MinScore

Type de donnée : num

Valeur réelle qui définit le seuil minimum de la tolérance « score » du traitement vision d'une caméra « In-Sight ». Une valeur inférieure sera considérée comme incorrecte.

Timeout

Type de donnée : num

Valeur réelle sous forme d'un paramètre optionnel qui définit le temps d'attente maximum pour une réponse de la caméra « In-Sight ». Par défaut, le temps d'attente est de 5 secondes.

**Remarque :**

Les 7 paramètres de seuil sont utilisés par la fonction booléenne « CX_GetVisionData » ; si un des paramètres est incorrect, cette fonction renverra « FALSE ».

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « Identity » doit être un entier compris entre 1 et 4,
- Le paramètre « SocketIP » doit être une chaîne correspondant à 4 entiers compris entre 0 et 255, séparés par un point,
- Le paramètre « SocketPort » doit être un entier positif,
- Le paramètre « MinScore » doit être une valeur réelle comprise entre 0 et 100,
- Le paramètre optionnel « Timeout » doit être une valeur réelle supérieure à 0.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_SetupCamera
[Identity' :='] <expression (IN) of num>',
[SocketIP' :='] <expression (IN) of string>',
[SocketPort' :='] <expression (IN) of num>',
[UserName' :='] <expression (IN) of string>',
[Password' :='] <expression (IN) of string>',
[MinCoordX' :='] <expression (IN) of num>',
[MaxCoordX' :='] <expression (IN) of num>',
[MinCoordY' :='] <expression (IN) of num>',
[MaxCoordY' :='] <expression (IN) of num>',
[MinCoordRZ' :='] <expression (IN) of num>',
[MaxCoordRZ' :='] <expression (IN) of num>',
[MinScore' :='] <expression (IN) of num>',
['\Timeout :='] <expression (IN) of num>' ;
```

4.1.2 Fonction « CX_InitComm »

CX_InitComm : initialisation de la communication avec une caméra

La fonction booléenne « CX_InitComm » permet d'initialiser la communication entre le robot et une caméra, précédemment configurée par l'instruction « CX_SetupCamera ».

Cette instruction doit être utilisée après l'instruction « CX_SetupCamera » et avant toutes les autres instructions de la librairie de communication (voir l'exemple de programmation au paragraphe 5.2).

Exemple

```
VAR num nError;  
  
IF CX_InitComm(1,nError) THEN  
    ...  
ELSE  
    ! Gestion d'erreur  
    ...  
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si la communication avec la camera numéro 1 est correctement établie. En cas d'échec d'initialisation de la communication, la fonction prend la valeur « **FALSE** » et la variable *nError* reçoit le code erreur correspondant.

Arguments

CX_InitComm(Identity,ErrStatus)

CameraId

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_InitComm '('  
[CameraId':=']<expression>(IN) of num>','  
[ErrStatus':=']<var or pers>(INOUT) of num>  
)'
```

4.1.3 Fonction « CX_TriggImage »

CX_TriggImage : déclenchement d'une prise de vue pour une caméra

La fonction booléenne « CX_TriggImage » permet le déclenchement d'une prise de vue pour la caméra spécifiée.

Cette instruction ne peut être utilisée que si la fonction « CX_InitComm » s'est correctement déroulée.

Exemple

```
VAR num nError;  
  
IF CX_TriggImage(1,nError) THEN  
    ! Traitement d'image  
...  
ELSE  
    ! Gestion d'erreur  
...  
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si la prise de vue avec la camera numéro 1 s'est correctement déroulée. En cas d'échec, elle prend la valeur « **FALSE** » et la variable *nError* reçoit le code erreur correspondant.

Arguments

CX_TriggImage(Identity,ErrStatus)

CameraId

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_TriggImage'( '  
[CameraId':=']<expression>(IN) of num>', '  
[ErrStatus':=']<var or pers>(INOUT) of num>  
)'
```

4.1.4 Fonction « CX_GetVisionData »

CX_GetVisionData : acquisition des données de vision pour une caméra

La fonction « CX_GetVisionData » permet de récupérer les coordonnées 2D (X, Y, RZ et score vision) d'un objet dans le programme vision d'une caméra « In-Sight ».

Il existe deux méthodes d'acquisition des coordonnées 2D :

- Récupération des coordonnées dans une seule cellule au format ABB,
- Récupération des coordonnées dans quatre cellules contiguës.

L'acquisition des coordonnées 2D dans une seule cellule est recommandée pour optimiser le temps de cycle de communication.

Exemple

```
PERS wobjdata wCam1:=[FALSE,TRUE,"",[...],[...]];
VAR pose pVision:=[0,0,0],[0,0,0,0];
VAR num nScore;
VAR num nError;

IF CX_GetVisionData(1,"F",22\Read1Cell,pVision,nScore,nError) THEN
    ! Utilisation du décalage vision
    wCam1.oframe:=pVision;
    ...
ELSE
    ! Gestion d'erreur
    ...
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si la cellule « F22 » du programme vision de la caméra 1 comporte une chaîne de caractères au format ABB :

- La variable *pVision* de type « pose » reçoit les coordonnées 2D du programme vision,
- La variable *nScore* de type « num » reçoit la valeur « score » du programme vision.

En cas d'échec, la fonction prend la valeur « **FALSE** » et la variable *nError* reçoit le code erreur correspondant.

Arguments

CX_GetVisionData(Identity,CellColumn,CellRow,[\Read1Cell][\Read4Cells],VisionTransform,
VisionScore,ErrStatus)

CameraId

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

CellColumn

Type de donnée : string

Chaîne de caractères qui définit la colonne de la cellule dans le programme vision, comprise entre A et Z.

CellRow

Type de donnée : num

Valeur entière qui définit la ligne de la cellule dans le programme vision, comprise entre 0 et 399.

\Read1Cell

Type de donnée : switch

Paramètre qui permet de lire les coordonnées 2D dans une seule cellule du programme vision définie au format ABB, à partir de la cellule définie par (**CellColumn,CellRow**) et défini par la syntaxe suivante :

`[ ' <x> ' , ' <y> ' , ' <rz> ' , ' <score> ' ]`



Remarque :

- Ce mode de lecture des coordonnées 2D est recommandé pour optimiser le temps de cycle de communication.
- Ce format correspond à une donnée composite du langage RAPID (identifié par le terme « aggregate » dans la documentation).
- Ce paramètre correspond au comportement par défaut de l'instruction quand aucun de ces 2 paramètres optionnels n'est spécifié par l'utilisateur.

\Read4Cells

Type de donnée : switch

Paramètre qui permet de lire les coordonnées 2D dans quatre cellules contiguës du programme vision à partir de la cellule définie par (**CellColumn,CellRow**) jusqu'à (**CellColumn+3,CellRow**).

Chaque cellule doit contenir une valeur réelle ou entière. Dans le cas contraire, la fonction renverra la valeur « **FALSE** ».

VisionTransform

Type de donnée : pose

Variable de type « pose » qui reçoit les coordonnées 2D mises en forme comme une transformation géométrique, directement utilisable pour déplacer un repère.

Le format de la transformation est « `[[X,Y,0],[Q1,0,0,Q4]]` » où :

- X et Y sont les coordonnées de translation du programme vision,
- Q1, Q2, Q3 et Q4 sont les « quaternions » calculés d'après l'angle de rotation RZ du programme vision. Dans ce cas particulier, les quaternions Q2 et Q3 sont toujours égaux à 0.

VisionScore

Type de donnée : num

Valeur réelle qui reçoit le score de reconnaissance de vision déterminée par la vision In-Sight.

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Programmation vision

Le formatage de la cellule du programme vision pour la lecture de toutes les coordonnées 2D en une seule cellule, peut être réalisé simplement avec la fonction « Concatenate » disponible en programmation de la caméra « In-Sight ». Exemple :

```
Concatenate ( " [ " , B22 , " , " , C22 , " , " , D22 , " , " , E22 , " ] " )
```

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.
- Le paramètre « CellColumn » doit être un caractère compris entre A et Z.
- Le paramètre « CellRow » doit être un entier compris entre 0 et 399.

Un message d'erreur est affiché si un des paramètres est incorrect.

Les coordonnées 2D et la valeur de « score » doivent être comprises dans les tolérances définies dans l'instruction *CX_SetupCamera*.

Syntaxe

```
CX_GetVisionData'(
[CameraId':=']<expression>(IN) of num>',
[CellColumn':=']<expression>(IN) of string>',
[CellRow':=']<expression>(IN) of num>',
['\Read1Cell','] |
['\Read4Cell',']
[VisionTransform':=']<var or pers>(INOUT) of pose>',
[VisionScore':=']<var or pers>(INOUT) of num>',
[ErrStatus':=']<var or pers>(INOUT) of num>
')
```

4.1.5 Instruction « CX_ShowErrStatus »

CX_ShowErrStatus : visualisation d'une erreur vision sur le pupitre du robot

L'instruction « CX_ShowErrStatus » permet d'afficher un message d'erreur sur l'écran du pupitre « FlexPendant ». Le message d'erreur est enregistré dans le journal d'évènements du système de commande IRC5.

Exemple

```
PERS wobjdata wCam1:=[FALSE, TRUE, "", [[0,0,0], [1,0,0,0]], [[0,0,0],
[1,0,0,0]]];
VAR pose pVision:=[[0,0,0], [0,0,0,0]];
VAR num nScore;
VAR num nError;

IF CX_GetVisionData(1, "F", 22\ReadlCell, pVision, nScore, nError) THEN
    ! Utilisation du décalage vision
    wCam1.oframe:=pVision;
...
ELSE
    ! Gestion d'erreur
    CX_ShowErrStatus(nError);
...
ENDIF
```

Si le résultat de la fonction CX_GetVisiondata est « **FALSE** », un message d'erreur sera affiché sur l'écran du pupitre et enregistré dans le journal d'évènements du système de commande IRC5.

Arguments

CX_ShowErrStatus(ErrStatus)

ErrStatus

Type de donnée : num

Valeur entière qui définit le numéro d'erreur à afficher.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « ErrStatus » doit être un entier.

En cas de numéro d'erreur inconnu, le message affiché sera de la forme :

« Erreur d'application : code d'erreur inconnu numéro <ErrStatus> »

Syntaxe

```
CX_ShowErrStatus
[ErrStatus':=']<var or pers>(INOUT) of num>';
```

4.1.6 Fonction « CX_CloseComm »

CX_CloseComm : fermeture de la communication avec une caméra

La fonction booléenne « CX_CloseComm » permet de fermer la communication entre le robot et une caméra, précédemment ouverte par l'instruction « CX_InitComm ».

Exemple

```
VAR num nError;  
  
IF CX_CloseComm(1,nError) THEN  
    ...  
ELSE  
    ! Gestion d'erreur  
    ...  
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si la communication avec la camera numéro 1 est correctement fermée. En cas de numéro de caméra incorrect, la fonction prend la valeur « **FALSE** » et la variable *nError* reçoit le code erreur correspondant.

Arguments

CX_CloseComm(Identity,ErrStatus)

CameraId

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_CloseComm '('  
[CameraId':=']<expression>(IN) of num>','  
[ErrStatus':=']<var or pers>(INOUT) of num>  
)'
```

4.1.7 Instruction « CX_SetLanguage »

CX_SetLanguage : définition de la langue de l'interface homme machine

L'instruction "CX_SetLanguage" permet de définir la langue de l'interface homme machine et de tous les messages générés par les instructions de communication (se référer au chapitre 8.2 pour plus de détails sur l'affichage).



Remarque :

L'Anglais est la langue utilisée par défaut si cette instruction n'est pas utilisée.

Exemple

```
CX_SetLanguage \German;
```

Arguments

CX_SetLanguage Dutch | English | French | German | Italian | Spanish

Dutch	Type de donnée : switch
Paramètre qui active la langue Hollandaise.	
English	Type de donnée : switch
Paramètre qui active la langue Anglaise (valeur par défaut).	
French	Type de donnée : switch
Paramètre qui active la langue Française.	
German	Type de donnée : switch
Paramètre qui active la langue Allemande.	
Italian	Type de donnée : switch
Paramètre qui active la langue Italienne.	
Spanish	Type de donnée : switch
Paramètre qui active la langue Espagnole.	
Swedish	Type de donnée : switch
Paramètre qui active la langue Suédoise.	

Syntaxe

```
CX_SetLanguage  
['\Dutch','] |  
['\English','] |  
['\French','] |  
['\German','] |  
['\Italian','] |  
['\Spanish'] |  
['\Swedish'] ;'
```

4.1.8 Instruction « CX_SetDebugMode »

CX_SetDebugMode : activation ou désactivation du mode « debug »

L'instruction "CX_SetDebugMode" permet d'activer ou de désactiver le mode « debug » pour faciliter la mise au point d'une application de communication. L'activation du mode « debug » affiche sur le « FlexPendant » l'intégralité des caractères échangés pendant la communication (se référer au chapitre 8.2 pour plus de détails sur l'affichage).

Exemple

```
CX_SetDebugMode \On;
```

Arguments

CX_SetDebugMode On|Off

On

Type de donnée : switch

Paramètre qui active le mode « debug ».

Off

Type de donnée : switch

Paramètre qui désactive le mode « debug ».

Limites

Cette instruction ne doit être utilisée que de façon **temporaire** pendant la mise au point d'une application.



Remarque :

Elle ne doit pas être utilisée pendant la production, car il y a un risque d'effacer ou de perturber la mise en page de l'affichage de textes dans la fenêtre de dialogue opérateur du pupitre « FlexPendant » réalisée en langage « RAPID » avec les instructions « TPWrite » ou « TPReadFK », qui sont souvent utilisées en production.

Syntaxe

```
CX_SetDebugMode  
['\On','] |  
['\Off','] ;'
```

4.1.9 Instruction « CX_ShowVersion »

CX_ShowVersion : affichage de la version de la librairie de communication

L'instruction "CX_ShowVersion" permet d'afficher sur le pupitre « FlexPendant » la version de la librairie de communication.

Cette fonction est nécessaire pour avoir accès à la version de la librairie de communication du fait de l'encryptage du code source. Cette fonction utilitaire n'est en principe pas utilisée pendant la production, mais uniquement en mode manuel pour information.

Exemple

```
CX_ShowVersion;
```

Arguments

CX_ShowVersion \Pause

Pause (argument optionnel)

Type de donnée : switch

Argument optionnel qui provoque une attente dans l'exécution du programme robot par la demande d'acquiescement de l'opérateur au pupitre « FlexPendant » (appuyer sur la touche « Ok »).

Si cet argument optionnel n'est pas utilisé, l'affichage de la version est réalisé dans la fenêtre « dialogue opérateur ».

Syntaxe

```
CX_ShowVersion  
['\Pause'] ;'
```

4.2 Instructions de lecture et d'écriture de données dans le programme « In-Sight »

Vue d'ensemble

Ce chapitre décrit les instructions suivantes:

- CX_GetInfo
- CX_GetValue
- CX_SetFloat
- CX_SetInteger
- CX_SetString

Ces instructions permettent de lire et d'écrire dans le programme vision d'une caméra « In-Sight ».

Se référer au paragraphe “Settings & Cell Values Commands” de la documentation COGNEX pour le détail du protocole utilisé.

4.2.1 Fonction « CX_GetInfo »

CX_GetInfo : lecture de l'état du système de vision

La fonction « CX_GetValue » permet de lire les données d'état du système de vision : le numéro de série, le numéro d'identité « adresse MAC », les versions et la date des logiciels systèmes.



Note :

La fonction « CX_GetInfo » n'est pas encore implémentée dans cette version et renvoie « FALSE ».

Exemple

```
IF NOT CX_GetInfo(1, strSN, strFWV, strMV, strMAC, strDoB, nError) THEN
...
ENDIF
```

Arguments

CX_GetInfo(CameraId, SerialNumber, FirmwareVersion, MonitorVersion, MacAddress, DateOfBuild, ErrStatus)

CameraId	Type de donnée : num
Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.	
SerialNumber	Type de donnée : string
Chaîne de caractères qui retourne le numéro de série de la caméra.	
FirmwareVersion	Type de donnée : string
Chaîne de caractères qui retourne la version du logiciel système « Firmware » de la caméra.	
MonitorVersion	Type de donnée : string
Chaîne de caractères qui retourne la version du logiciel système « Monitor » de la caméra.	
MacAddress	Type de donnée : string
Chaîne de caractères qui retourne le numéro « MAC » du port Ethernet de la caméra.	
DateOfBuild	Type de donnée : string
Chaîne de caractères qui retourne la date de construction du logiciel système « Firmware » de la caméra.	
ErrStatus	Type de donnée : num
Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.	

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_GetInfo'(
[CameraId':=']<expression>(IN) of num>',
[SerialNumber':=']<var or pers>(INOUT) of string>',
[FirmwareVersion':=']<var or pers>(INOUT) of string>',
[MonitorVersion':=']<var or pers>(INOUT) of string>',
[MacAddress':=']<var or pers>(INOUT) of string>',
[DateOfBuild':=']<var or pers>(INOUT) of string>',
[ErrStatus':=']<var or pers>(INOUT) of num>
')
```

4.2.2 Fonction « CX_GetValue »

CX_GetValue : lecture d'une chaîne de caractères dans le programme vision

La fonction « CX_GetValue » permet de lire la valeur d'une cellule du programme vision.

Exemple

```
VAR num nError;  
VAR string strValue;  
  
IF NOT CX_GetValue(1, "C", 29, strValue, nError) THEN  
    CX_ShowErrStatus(nError) ;  
    RETURN;  
ENDIF
```

La fonction retourne la valeur « **TRUE** » si la chaîne de caractères « strValue » a pu prendre la valeur du contenu de la cellule « C29 ». Sinon elle retourne la valeur « **FALSE** » et le numéro d'erreur dans « nError ».

Arguments

CX_GetValue(Identity, CellColumn, CellRow, StringData, ErrStatus)

CameraId	Type de donnée : num
Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.	
CellColumn	Type de donnée : string
Chaîne de caractères qui définit la colonne à pointer programme vision, compris entre A et Z.	
CellRow	Type de donnée : num
Valeur entière qui définit la ligne à pointer du programme vision, compris entre 0 et 399.	
StringData	Type de donnée : string
Chaîne de caractères qui retourne la valeur de la cellule spécifiée du programme vision.	
ErrStatus	Type de donnée : num
Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.	

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.
- Le paramètre « CellColumn » doit être un caractère compris entre A et Z.
- Le paramètre « CellRow » doit être un entier compris entre 0 et 399.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_GetValue'(
[CameraId':=']<expression>(IN) of num>',
[CellColumn':=']<expression>(IN) of string>',
[CellRow':=']<expression>(IN) of num>',
[StringData':=']<expression>(INOUT) of string>',
[ErrStatus':=']<var or pers>(INOUT) of num>
```

4.2.3 Fonction « CX_SetFloat »

CX_SetFloat : écriture d'une valeur réelle dans le programme vision

La fonction « CX_SetFloat » permet d'écrire une valeur réelle dans une cellule du programme vision d'une caméra « In-Sight ».

Exemple

```
VAR num nError;  
  
IF NOT CX_SetFloat(1,"C",29,451.32,nError) THEN  
    CX_ShowErrStatus(nError);  
    RETURN;  
ENDIF
```

La fonction va chercher à écrire le nombre 451,32 dans la cellule « C29 » du programme vision. Si l'opération réussit, la fonction renvoie la valeur « **TRUE** ».

La cellule destination doit être au bon format, sinon la fonction retournera la valeur « **FALSE** » et le numéro d'erreur dans « nError ».

Arguments

CX_SetFloat(Identity,CellColumn,CellRow,FloatData,ErrStatus)

CameraId	Type de donnée : num
Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.	
CellColumn	Type de donnée : string
Chaîne de caractères qui définit la colonne à pointer du programme vision, compris entre A et Z.	
CellRow	Type de donnée : num
Valeur entière qui définit la ligne à pointer du programme vision, compris entre 0 et 399.	
FloatData	Type de donnée : num
Valeur réelle qui est écrite dans la cellule spécifiée du programme vision.	
ErrStatus	Type de donnée : num
Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.	

Programmation vision

La cellule du programme vision doit être définie avec un contrôle interactif de type « EditFloat ». Les valeurs minimum et maximum définies par défaut (respectivement 0 et 100) doivent être ajustées en fonction de la valeur réelle envoyée par le robot.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un nombre entier compris entre 1 et 4,
- Le paramètre « CellColumn » doit être un caractère compris entre A et Z,
- Le paramètre « CellRow » doit être un nombre entier compris entre 0 et 399,
- Le paramètre « FloatData » doit être un nombre réel ou entier.

Un message d'erreur est affiché si un des paramètres est incorrect.

La caméra « In-Sight » renvoie une erreur si la cellule du programme vision ne contient pas un contrôle interactif de type « EditFloat » ou si le nombre envoyé par le robot n'est pas compris dans les limites minimum et maximum du contrôle.

Syntaxe

```
CX_SetFloat'(
[CameraId':=']<expression>(IN) of num>',
[CellColumn':=']<expression>(IN) of string>',
[CellRow':=']<expression>(IN) of num>',
[FloatData':=']<var or pers>(IN) of num>',
[ErrStatus':=']<var or pers>(INOUT) of num>
')
```

4.2.4 Fonction « CX_SetInteger »

CX_SetInteger : écriture d'une valeur entière dans le programme vision

La fonction « CX_SetInteger » permet d'écrire une valeur entière dans une cellule du programme vision.

Exemple

```
IF NOT CX_SetInteger(1,"C",29,221,nError) THEN
    CX_ShowErrStatus(nError);
    RETURN;
ENDIF
```

La fonction va chercher à écrire le nombre entier 221 dans la cellule « C29 » du programme vision. Si l'opération réussit, la fonction renvoie la valeur « **TRUE** ».

La cellule destination doit être au bon format, sinon la fonction retournera la valeur « **FALSE** » et le numéro d'erreur dans « nError ».

Arguments

CX_SetInteger(Identity,CellColumn,CellRow,IntegerData,ErrStatus)

CameraId Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

CellColumn Type de donnée : string

Chaîne de caractères qui définit la colonne à pointer du programme vision, compris entre A et Z.

CellRow Type de donnée : num

Valeur entière qui définit la ligne à pointer du programme vision, compris entre 0 et 399.

IntegerData Type de donnée : num

Valeur entière qui est écrite dans la cellule spécifiée du programme vision, comprise entre 0 et 255.

ErrStatus Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Programmation vision

La cellule du programme vision doit être définie avec un contrôle interactif de type « EditInt », « CheckBox » ou « ListBox ». Les valeurs minimum et maximum définies par défaut pour chaque type de contrôle doivent être ajustées en fonction de la valeur réelle envoyée par le robot :

- « EditInt » : minimum 0 et maximum 255,
- « CheckBox » : minimum 0 et maximum 1,
- « ListBox » : minimum 0 et maximum fonction du nombre d'éléments de la liste.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un nombre entier compris entre 1 et 4.
- Le paramètre « CellColumn » doit être un caractère compris entre A et Z.
- Le paramètre « CellRow » doit être un nombre entier compris entre 0 et 399.
- Le paramètre « IntegerData » doit être un nombre entier.

Un message d'erreur est affiché si un des paramètres est incorrect.

La caméra « In-Sight » renvoie une erreur si la cellule du programme vision ne contient pas un contrôle interactif de type « EditInt » ou si le nombre envoyé par le robot n'est pas compris dans les limites minimum et maximum du contrôle.

Syntaxe

```
CX_SetInteger'(  
[CameraId':=']<expression>(IN) of num>',  
[CellColumn':=']<expression>(IN) of string>',  
[CellRow':=']<expression>(IN) of num>',  
[IntegerData':=']<var or pers>(IN) of num>',  
[ErrStatus':=']<var or pers>(INOUT) of num>  
)'
```

4.2.5 Fonction « CX_SetString »

CX_SetString : écriture d'une chaîne de caractères dans le programme vision

La fonction « CX_SetString » permet d'écrire une chaîne de caractères dans une cellule du programme vision.

Exemple

```
IF NOT CX_SetString(1,"C",29,"ABB",nError) THEN
    CX_ShowErrStatus(nError) ;
    RETURN;
ENDIF
```

La fonction va chercher à écrire la chaîne « ABB » dans la cellule « C29 » du programme vision. Si l'opération réussit, la fonction renvoie la valeur « **TRUE** ».

La cellule destination doit être au bon format, sinon la fonction retournera la valeur « **FALSE** » et le numéro d'erreur dans « nError ».

Arguments

CX_SetString(Identity,CellColumn,CellRow,StringData,ErrStatus)

CameraId

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

CellColumn

Type de donnée : string

Chaîne de caractères qui définit la colonne à pointer du programme vision, compris entre A et Z.

CellRow

Type de donnée : num

Valeur entière qui définit la ligne à pointer du programme vision, compris entre 0 et 399.

StringData

Type de donnée : string

Chaîne de caractères qui est écrite dans la cellule spécifiée du programme vision.

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Programmation vision

La cellule du programme vision doit être définie avec un contrôle interactif de type « EditString ». La longueur maximale de la chaîne de caractères définie par défaut (8) doit être ajustée en fonction de la valeur réelle envoyée par le robot.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.
- Le paramètre « CellColumn » doit être un caractère compris entre A et Z.
- Le paramètre « CellRow » doit être un entier compris entre 0 et 399.

Un message d'erreur est affiché si un des paramètres est incorrect.

La caméra « In-Sight » renvoie une erreur si la cellule du programme vision ne contient pas un contrôle interactif de type « EditString » ou si la chaîne de caractères envoyée par le robot est trop longue (c.à.d. dépasse la taille maximale définie dans le contrôle).

Syntaxe

```
CX_SetString'( '  
[CameraId':=']<expression>(IN) of num>', '  
[CellColumn':=']<expression>(IN) of string>', '  
[CellRow':=']<expression>(IN) of num>', '  
[StringData':=']<expression>(IN) of string>', '  
[ErrStatus':=']<var or pers>(INOUT) of num>  
)'
```

4.3 Instructions d'exécution et de gestion du mode « Online »

Vue d'ensemble

Ce chapitre décrit les instructions suivantes:

- CX_GetOnline
- CX_SendMessage
- CX_SetEvent
- CX_SetEventWait
- CX_SetOnline

Ces instructions permettent de gérer les événements pendant l'exécution du programme d'une caméra « In-Sight », ainsi que le mode « Online ».

Se référer au paragraphe “ Execution & Online Commands ” de la documentation COGNEX pour le détail du protocole utilisé.

4.3.1 Fonction « CX_GetOnline »

CX_GetOnLine : vérification de l'état "en ligne" pour une caméra

La fonction booléenne « CX_GetOnline » permet la vérification de l'état en ligne pour la caméra spécifiée.

Cette instruction ne peut être utilisée que si la fonction « CX_InitComm » s'est correctement déroulée.

Exemple

```
VAR num nError;  
  
IF CX_GetOnline(1,nError) THEN  
    ! Traitement d'image  
    ...  
ELSE  
    ! Gestion d'erreur  
    ...  
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si la camera numéro 1 est en ligne. Dans le cas contraire, la fonction retournera la valeur « **FALSE** » et le numéro d'erreur dans « nError ».

Arguments

CX_GetOnLine(Identity,ErrStatus)

CameraId

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_GetOnLine'( '  
[CameraId':=']<expression>(IN) of num>', '  
[ErrStatus':=']<var or pers>(INOUT) of num>  
)'
```

4.3.2 Fonction « CX_SendMessage »

CX_SendMessage : envoi d'un message au programme vision

La fonction booléenne « CX_SendMessage » permet l'envoi d'un message et d'un code évènement optionnel au programme vision.

Exemple

```
VAR num nError;  
  
IF CX_SendMessage(1,"ABB",\EventNb:=1,nError) THEN  
    ! Traitement  
    ...  
ELSE  
    ! Gestion d'erreur  
    ...  
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si l'envoi de l'évènement sur la camera numéro 1 s'est correctement déroulé. En cas d'échec, elle prend la valeur « **FALSE** » et la variable *nError* reçoit le code erreur correspondant.

Arguments

CX_SendMessage(Identity,ErrStatus)

CameraId

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

Message

Type de donnée : string

Chaîne de caractères qui définit le message à envoyer au programme vision.

\EventNb

Type de donnée : num

Valeur entière optionnelle qui définit le numéro d'évènement à envoyer au programme vision.

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.
- Le paramètre « EventNb » doit être un entier dans la plage reconnue par le RAPID.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_SendMessage'(
[CameraId':=']<expression>(IN) of num>',
[Message':=']<expression>(IN) of string>',
[EventNb':=']<expression>(IN) of num>',
[ErrStatus':=']<var or pers>(INOUT) of num>
')
```

4.3.3 Fonction « CX_SetEvent »

CX_SetEvent : envoi d'un code évènement au programme de vision

La fonction booléenne « CX_SetEvent » permet l'envoi d'un code évènement au programme vision. Le robot n'attend **pas** la fin du traitement de cet évènement pour poursuivre son programme.

Exemple

```
VAR num nError;

IF CX_SetEvent(1,101,nError) THEN
    ! Traitement
    ...
ELSE
    ! Gestion d'erreur
    ...
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si l'envoi de l'évènement sur la caméra numéro 1 s'est correctement déroulé. Dans le cas contraire, la fonction retournera la valeur « **FALSE** » et le numéro d'erreur dans « nError ».

Arguments

CX_SetEvent(CameraId,EventNb,ErrStatus)

CameraId	Type de donnée : num
Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.	
EventNb	Type de donnée : num
Valeur entière qui définit le numéro d'évènement à envoyer au programme vision.	
ErrStatus	Type de donnée : num
Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.	

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.
- Le paramètre « EventNb » doit être un entier dans la plage reconnue par le RAPID.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_SentEvent('
[CameraId':=']<expression>(IN) of num>',
[EventNb':=']<expression>(IN) of num>',
[ErrStatus':=']<var or pers>(INOUT) of num>
```

’)

4.3.4 Fonction « CX_SetEventWait »

CX_SetEventWait : envoi d'un code événement au programme de vision et attente de la fin de traitement

La fonction booléenne « CX_SetEventWait » permet l'envoi d'un code événement au programme de vision. Le robot attend la fin du traitement de cet événement pour poursuivre son programme.

Exemple

```
VAR num nError;

IF CX_SetEventWait(1,101,nError) THEN
    ! Traitement
    ...
ELSE
    ! Gestion d'erreur
    ...
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si l'envoi de l'évènement sur la camera numéro 1 s'est correctement déroulé. Dans le cas contraire, la fonction retournera la valeur « **FALSE** » et le numéro d'erreur dans « nError ».

Arguments

CX_SendEventWait(CameraId,EventNb,ErrStatus)

CameraId	Type de donnée : num
Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.	
EventNb	Type de donnée : num
Valeur entière qui définit le numéro d'évènement à envoyer au programme vision.	
ErrStatus	Type de donnée : num
Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.	

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.
- Le paramètre « EventNb » doit être un entier dans la plage reconnue par le RAPID.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_SendEventWait('
[Identity':=']<expression>(IN) of num>',
[EventNb':=']<expression>(IN) of num>',
[ErrStatus':=']<var or pers>(INOUT) of num>
```

’)

4.3.5 Fonction « CX_SetOnline »

CX_SetOnline : mise en état "en ligne" pour une caméra

La fonction booléenne « CX_GetOnline » permet la mise en ligne ou hors-ligne pour la caméra spécifiée.

Cette fonction ne peut être utilisée que si la fonction « CX_InitComm » s'est correctement déroulée.

Exemple

```
VAR num nError;  
  
IF CX_SetOnline(1\On,nError) THEN  
    ! Traitement d'image  
    ...  
ELSE  
    ! Gestion d'erreur  
    ...  
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si la mise en ligne de la camera numéro 1 s'est correctement déroulée. Dans le cas contraire, la fonction retournera la valeur « **FALSE** » et le numéro d'erreur dans « nError ».

Arguments

CX_SetOnline(Identity,\OnOff,ErrStatus)

CameraId

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

\On

Type de donnée : switch

Permet la mise en ligne de la caméra.

\Off

Type de donnée : switch

Permet la mise hors ligne de la caméra. Complément du switch \On, c'est la valeur par défaut lorsque aucun des deux switches n'est spécifié.

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_SetOnline'(  
[CameraId':=']<expression>(IN) of num>',  
['\On','] |  
['\Off',']  
[ErrStatus':=']<var or pers>(INOUT) of num>  
)'
```

4.4 Instructions de gestion de fichiers

Vue d'ensemble

Ce chapitre décrit les instructions suivantes:

- CX_GetFile
- CX_LoadFile

Ces instructions permettent de gérer les fichiers programme d'une caméra « In-Sight ».

Se référer au paragraphe “ File & Job Commands ” de la documentation COGNEX pour le détail du protocole utilisé.

4.4.1 Fonction « CX_GetFile »

CX_GetFile : lecture du nom du programme vision chargé dans la caméra

La fonction booléenne « CX_GetFile » permet de lire le nom du programme vision chargé dans la mémoire d'exécution de la caméra « In-Sight ».

Exemple

```
VAR string strFileName;  
VAR num nError;  
  
IF CX_GetFile(1,strFileName,nError) THEN  
    ! Traitement  
    ...  
ELSE  
    ! Gestion d'erreur  
    ...  
ENDIF
```

La fonction renvoie la valeur « **TRUE** » si le programme vision de la camera numéro 1 a été sauvegardé ; le nom du programme vision est alors retourné dans la variable « strFileName ».

Si aucun programme vision n'est chargé, elle prend la valeur « **FALSE** ».

En cas d'échec, elle prend la valeur « **FALSE** » et la variable *nError* reçoit le code erreur correspondant.

Arguments

CX_GetFile(CameraId,FileName,ErrStatus)

CameraId Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

FileName Type de donnée : string

Chaîne de caractères qui retourne le nom du programme vision chargé dans la caméra « In-Sight ».

ErrStatus Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_GetFile'(
[CameraId':=']<expression>(IN) of num>',
[FileName':=']<var or pers>(INOUT) of string>',
[ErrStatus':=']<var or pers>(INOUT) of num>
')
```

4.4.2 Fonction « CX_LoadFile »

CX_LoadFile : chargement d'un programme vision dans la caméra

La fonction booléenne « CX_LoadFile » permet de charger un programme vision (« job ») de la mémoire flash de la caméra « In-Sight » vers sa mémoire d'exécution.

**Note :**

Cette fonction ne peut être utilisée que si la caméra a été au préalable mise « hors-ligne » par le robot, en utilisant la fonction « CX_SetOnline ». La caméra doit être remise « en ligne » de la même façon après l'exécution de la fonction « CX_LoadFile ».

Exemple

```
VAR string strFileName;  
VAR num nError;  
  
IF NOT CX_SetOnline(1\Off,nError) Exit;  
IF NOT CX_LoadFile(1,strFileName,nError) THEN  
    ! Gestion d'erreur  
    ...  
ENDIF  
IF NOT CX_SetOnline(1\On,nError) Exit;
```

Arguments

CX_LoadFile(Identity,ErrStatus)

CameraId

Type de donnée : num

Valeur entière qui définit le numéro d'identité d'une caméra, compris entre 1 et 4.

FileName

Type de donnée : string

Chaîne de caractères qui définit le nom du programme vision à charger dans la caméra « In-Sight ». L'extension « .job » peut être omise (la caméra rajoute l'extension dans ce cas).

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « CameraId » doit être un entier compris entre 1 et 4.

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
CX_LoadFile(''  
[CameraId':='<expression>(IN) of num>','  
[FileName':='<var or pers>(IN) of string>','  
[ErrStatus':='<var or pers>(INOUT) of num>  
'')
```

4.5 Instructions d'étalonnage robot-vision

Vue d'ensemble

Ce chapitre décrit les instructions suivantes :

- SetToolVision
- SetWObjVision

Ces instructions permettent de gérer l'étalonnage robot-vision dans le cas d'une caméra fixe et d'une caméra embarquée sur le robot.

Se référer au chapitre 7 "Procédures d'étalonnage robot-vision" de cette documentation pour le détail de la mise en œuvre de ces instructions.

4.5.1 Instruction « SetToolVision »

SetToolVision : calcul de l'étalonnage robot-vision pour caméra embarquée sur le robot

L'instruction « SetToolVision » permet de calculer la transformation de l'étalonnage robot-vision dans le cas d'une caméra embarquée sur le robot et renvoie un repère outil associé à la caméra.



Note :

La fonction « SetToolVision » n'est pas encore implémentée dans cette version de l'interface. Contactez ABB France division Robotique pour le support technique.

Exemple

```
VAR robtarget RobotPos{4};
VAR pose VisionPos{4};
PERS tooldata toolVision;
VAR num nError;

SetToolVision RobotPos,VisionPos,4,\FRAME_2D,toolVision,nError;
```

L'instruction renvoie le repère outil *toolVision* associé à la caméra si le calcul d'étalonnage est correct. En cas d'échec, la variable *nError* reçoit le code erreur correspondant.

Arguments

SetToolVision RobotPos,VisionPos,toolVision,ErrStatus

RobotCalPos

Type de donnée : robtarget

Tableau de positions robot qui définit les positions d'étalonnage apprises dans le repère de base ou le repère atelier du robot.

VisionCalPos

Type de donnée : pose

Tableau de transformations vision qui définit les positions d'étalonnage apprises dans le repère vision.

NbCalPos

Type de donnée : num

Valeur entière qui définit le nombre de positions d'étalonnage, compris entre 4 et 20.

FRAME_2D (argument optionnel)

Type de donnée : switch

Argument optionnel qui spécifie que le repère outil sera calculé en 2D à partir d'un repère nominal, c.à.d. les composantes X/Y/RZ dans le repère vision ; 4 positions sont nécessaires. Cette méthode est utilisée par défaut si aucun argument optionnel n'est programmé.

FRAME_3D (argument optionnel)

Type de donnée : switch

Argument optionnel qui spécifie que toutes les composantes 3D du repère outil seront calculées. Cet argument nécessite la mise en œuvre d'une mire multipoints (non encore implémenté).

Tool

Type de donnée : tooldata

Repère outil dans lequel l'instruction retourne la transformation associée à la caméra « In-Sight ».

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « NbCalPos » doit être un entier compris entre 4 et 20.
- Les tableaux « RobotCalPos » et « VisionCalPos » doivent contenir le nombre requis d'éléments (4 à 20 selon l'argument optionnel choisi).

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
SetToolVision'( '  
[RobotCalPos':=']<expression>(IN) of robtarget>', '  
[VisionCalPos':=']<expression>(IN) of pose>', '  
['\FRAME_2D', '  
['\FRAME_3D', '  
[Tool:=']<pers>(INOUT) of tooldata>', '  
[ErrStatus':=']<VAR or PERS>(INOUT) of num>  
)'
```

4.5.2 Instruction « SetWObjVision »

SetWObjVision : calcul de l'étalonnage robot-vision pour caméra fixe

L'instruction « SetWObjVision » permet de calculer la transformation de l'étalonnage robot-vision dans le cas d'une caméra fixe et renvoie un repère objet associé à la caméra.

Exemple

```
VAR robtarget RobotPos{9};
VAR pose VisionPos{9};
PERS wobjdata wobjVision;
VAR num nError;

SetWObjVision RobotPos, VisionPos, 9, wobjVision, nError;
```

L'instruction renvoie le repère objet *wobjVision* associé à la caméra si le calcul d'étalonnage est correct. En cas d'échec, la variable *nError* reçoit le code erreur correspondant.

Arguments

SetWObjVision RobotPos, VisionPos, 9, wobjVision, ErrStatus

RobotCalPos

Type de donnée : robtarget

Tableau de positions robot qui définit les positions d'étalonnage apprises dans le repère de base ou le repère atelier du robot.

VisionCalPos

Type de donnée : pose

Tableau de transformations vision qui définit les positions d'étalonnage apprises dans le repère vision.

NbCalPos

Type de donnée : num

Valeur entière qui définit le nombre de positions d'étalonnage, compris entre 3 et 20.

UFrame (argument optionnel)

Type de donnée : switch

Argument optionnel qui retourne la transformation d'étalonnage dans la composante « uframe » du repère objet (utilisé par défaut si aucun argument optionnel n'est programmé).

OFrame (argument optionnel)

Type de donnée : switch

Argument optionnel qui retourne la transformation d'étalonnage dans la composante « oframe » du repère objet.

WObj

Type de donnée : wobjdata

Repère objet dans lequel l'instruction retourne la transformation associée à la caméra « In-Sight ».

MaxError (argument optionnel)

Type de donnée : num

Argument optionnel qui retourne l'erreur maximale du calcul d'étalonnage.

MeanError (argument optionnel)

Type de donnée : num

Argument optionnel qui retourne l'erreur moyenne du calcul d'étalonnage.

ErrStatus

Type de donnée : num

Valeur entière dans laquelle la fonction retourne un code erreur le cas échéant.

Limites

L'instruction vérifie les paramètres suivants fournis par l'utilisateur :

- Le paramètre « NbCalPos » doit être un entier compris entre 3 et 10.
- Les tableaux « RobotCalPos » et « VisionCalPos » doivent contenir au minimum le nombre d'éléments spécifiés dans le paramètre « NbCalPos ».

Un message d'erreur est affiché si un des paramètres est incorrect.

Syntaxe

```
SetWObjVision'( '  
[RobotCalPos':=']<expression>(IN) of robtarget>', '  
[VisionCalPos':=']<expression>(IN) of pose>', '  
[NbCalPos':=']<expression>(IN) of num>', '  
['\UFrame', '  
['\OFrame', '  
[WObj:=']<pers>(INOUT) of wobjdata>', '  
['\MaxError':=']<expression>(INOUT) of num>', '  
['\MeanError':=']<expression>(INOUT) of num>', '  
[ErrStatus':=']<VAR or PERS>(INOUT) of num>  
)'
```

5 Mise en œuvre des instructions de la librairie de communication

5.1 Séquences type d'utilisation des instructions de la librairie

Vue d'ensemble

L'exploitation la plus simple de la librairie de communication est fournie par la séquence suivante des instructions détaillées dans le chapitre 4 :

- | | |
|---------------------|---|
| 1. CX_SetLanguage | Définition de la langue de l'utilisateur |
| 2. CX_SetupCamera | Configuration de la caméra |
| 3. CX_InitComm | Initialisation de la communication avec la caméra |
| 4. CX_TriggImage | Demande de prise de vue à par la caméra |
| 5. CX_GetVisionData | Lecture des coordonnées dans la caméra |
| 6. CX_CloseComm | Fermeture de la communication avec la caméra |

Cette séquence est propre à une caméra et doit être répétée pour chaque caméra utilisée avec le système de commande IRC5.

Cas d'une caméra

Dans le cas d'une caméra, seul le déclenchement de la prise de vue et la lecture des données de vision sont utilisés dans la boucle de production :

```
CX_SetLanguage \French;      Définition de la langue utilisateur
CX_SetupCamera 1,...         Configuration de la caméra
CX_InitComm 1,...            Initialisation de la communication
...
WHILE <Production> DO
    ...
    CX_TriggImage 1,...       Demande de prise de vue à par la caméra
    ...
    CX_GetVisionData 1,...    Lecture des coordonnées dans la caméra
    ...
ENDWHILE
...
CX_CloseComm 1,...           Fermeture de la communication
```

Cette séquence élémentaire d'instructions peut être complétée par des cycles de lecture et d'écriture de données dans le programme de vision, typiquement pour paramétrer le type de pièce en cours de production. Les instructions les plus utilisées sont les suivantes :

- « CX_SetFloat » Ecriture d'une valeur réelle dans le programme vision,
- « CX_SetInteger » Ecriture d'une valeur entière dans le programme vision,
- « CX_SetString » Ecriture d'une chaîne de caractères dans le programme vision.

D'autres services peuvent également être ajoutés pour lire par exemple l'état de la caméra avant de lancer la production (« CX_GetOnline »).

Cas de plusieurs caméras

Dans le cas de plusieurs caméras, une limite de la version actuelle de la librairie impose une séquence d'instruction différente ; l'initialisation et la fermeture de la communication doivent être situées dans la boucle de production, avec le déclenchement de la prise de vue et la lecture des données de vision :

```
CX_SetLanguage \French;      Définition de la langue utilisateur
CX_SetupCamera 1,...         Configuration de la caméra 1
CX_SetupCamera 2,...         Configuration de la caméra 2
...
WHILE <Production> DO
    ...
    CX_InitComm 1,...         Initialisation communication caméra 1
    CX_TriggerImage 1,...     Demande de prise de vue à la caméra 1
    ...
    CX_GetVisionData 1,...    Lecture des coordonnées caméra 1
    CX_CloseComm 1,...        Fermeture communication caméra 1
    ...
    CX_InitComm 2,...         Initialisation communication caméra 2
    CX_TriggerImage 2,...     Demande de prise de vue à la caméra 2
    ...
    CX_GetVisionData 2,...    Lecture des coordonnées caméra 2
    CX_CloseComm 2,...        Fermeture communication caméra 2
    ...
ENDWHILE
...
```



Remarque :

Cette limitation sera supprimée dans une prochaine version.

5.2 Utilisation de la transformation de vision

Vue d'ensemble

L'instruction «CX_GetVisionData» retourne une transformation calculée à partir des coordonnées 2D fournies par la caméra «In-Sight». Une transformation peut être directement utilisée dans le langage de programmation «RAPID» du système de commande IRC5 de façons suivantes :

- Dans une instruction de déplacement de programme (cas d'une caméra fixe),
- Dans la donnée de repère objet (cas d'une caméra fixe),
- Dans un calcul de transformation défini par l'utilisateur.

Repère caméra et transformation de vision

La transformation de vision est exprimée dans le repère caméra. Pour être correctement exploitée dans le programme robot, il est impératif d'avoir réalisé au préalable :

- La procédure d'étalonnage du plan image réalisé au niveau de la vision pour prendre en compte le rapport pixels/mm ou les éventuelles déformations optiques. Voir le manuel d'utilisation de COGNEX, en particulier l'assistant «EasyBuilder» qui inclut maintenant une procédure automatique pour gérer cet étalonnage,
- La procédure d'étalonnage robot/vision (voir détails au chapitre 7). L'opération d'étalonnage robot/vision permet de mettre en corrélation le repère de la caméra et un repère utilisateur du robot (repère objet ou repère outil selon que la caméra est fixe au sol ou embarquée sur le robot).

Chaînage des repères géométriques utilisés pendant l'exploitation (cas d'une caméra fixe)

Le chaînage des repères géométriques utilisé pendant l'exploitation est défini par la figure suivante :

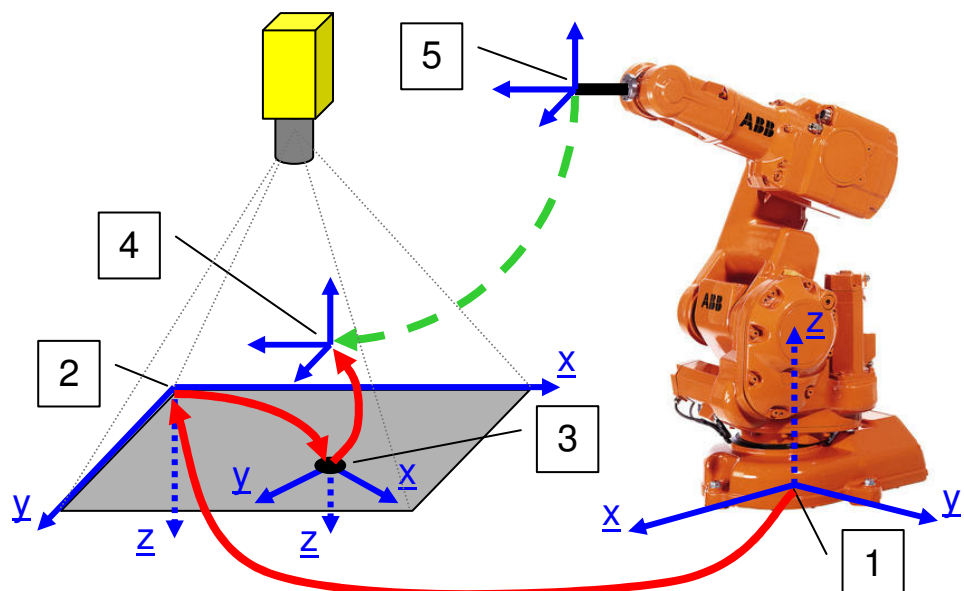


Figure 2. *Chaînage de repères géométriques utilisé pendant l'exploitation*

1	Repère de base du robot
2	Repère objet du robot (confondu avec le repère caméra après l'étalonnage robot/vision)
3	Repère du déplacement du programme, défini par la transformation de vision
4	Repère de la position à atteindre par le robot, défini par l'instruction de mouvement
5	Repère outil du robot

Mise en œuvre dans une instruction de déplacement de programme (cas d'une caméra fixe)

L'instruction de déplacement de programme est « PDispSet » (se référer au manuel de référence du langage « RAPID » pour plus de détails).

Exemple d'utilisation de la transformation de vision dans un déplacement de programme :

```
VAR pose pVision;  
...  
IF CX_GetVisionData(1, "F", 22, pVision, nScore, nError) THEN  
    ! Utilisation dans un déplacement de programme  
    PDispSet pVision;  
    ...  
    ! Mouvements du robot par rapport à l'objet de vision  
    ...  
    ! Réinitialisation du déplacement de programme  
    PDispOff;  
ELSE  
    ! Gestion d'erreur  
    ...  
ENDIF
```

L'instruction « PDispSet » est utilisée pour réaliser un décalage géométrique temporaire du programme et est donc bien adaptée à l'exploitation des coordonnées d'un objet fournies par une caméra, qui vont par principe changer à chaque cycle de production.



Remarque :

Le déplacement de programme est réinitialisé à une transformation nulle avec :

- L'instruction « PDispOff »,
- La réinitialisation du programme robot (par exemple, avec un déplacement manuel du pointeur d'exécution au début du programme ou un chargement d'un nouveau programme).



Attention :

Il faut faire attention à l'utilisation du déplacement de programme, car il peut être réinitialisé facilement, en particulier lors d'opérations en mode manuel. La réinitialisation d'un déplacement de programme peut provoquer des déplacements imprévus du robot.

Mise en œuvre dans une donnée de repère objet (cas d'une caméra fixe)

Un repère objet est défini en langage « RAPID » par le type « wobjdata ». Ce type de donnée structurée comprend plusieurs composantes dont deux transformations chaînées « uframe » et « oframe » :

- La composante « uframe » sera utilisée pour stocker la transformation définie à l'issue de l'étalonnage robot/vision (voir détails au chapitre 7),
- La composante « oframe » sera utilisée pour stocker la transformation vision associée à la position courante de l'objet.

Exemple d'utilisation de la transformation de vision dans une donnée de repère objet :

```
VAR pose pVision;  
! Déclaration d'une transformation nulle  
CONST pose pTRANSF0:=[[0,0,0],[1,0,0,0]];  
! Déclaration de la transformation vision  
VAR pose pVision:=[[0,0,0],[1,0,0,0]];  
...  
IF CX_GetVisionData(1,"F",22,pVision,nScore,nError) THEN  
    ! Utilisation dans une donnée de repère objet  
    wobjCamera.oframe:=pVision;  
    ...  
    ! Mouvements du robot par rapport à l'objet de vision  
    ...  
    ! Réinitialisation du repère objet  
    wobjCamera.oframe:=pTRANSF0;  
ELSE  
    ! Gestion d'erreur  
    ...  
ENDIF
```

Limitation : cette méthode n'est utilisable que si le repère objet n'est pas associé au mouvement d'une unité mécanique (par exemple un axe externe rotatif) ou un convoyeur. Dans le cas contraire, la composante « uframe » est réservée par le système de commande pour stocker en temps réel les coordonnées du repère objet déplacé par l'unité mécanique ou l'axe externe. Utilisez alors la méthode du déplacement de programme.

Mise en œuvre dans un calcul de transformation défini par l'utilisateur (cas d'une caméra fixe)



Remarque :

Cette troisième méthode s'adresse aux **programmeurs expérimentés**. La connaissance des fonctions de calcul suivantes du langage RAPID est indispensable pour mettre en œuvre cette méthode :

- « PoseInv »,
- « PoseMult »,
- « PoseVect ».

Une utilisation possible consiste à recalculer la position de prise de la pièce par le robot à partir de la position de prise nominale de la pièce (c.à.d. sans décalage de l'objet). Exemple d'utilisation de la transformation de vision dans un calcul de transformation défini par l'utilisateur :

```
! Déclaration de la transformation vision
VAR pose pVision;
! Déclaration de la transformation de calcul temporaire
VAR pose pTemp;
! Déclaration de la position de prise
CONST robtarget pPrise:=[[...], [...], [...], [...]];
VAR robtarget pPriseVision:=[[...], [...], [...], [...]];
...
IF CX_GetVisionData(1, "F", 22, pVision, nScore, nError) THEN
    ! Initialisation de la position de prise modifiée
    pPriseVision:=pPrise;
    ! Initialisation de la transformation temporaire
    pTemp.trans:=pPrise.trans;
    pTemp.rot:=pPrise.rot;
    ! Calcul de la position de prise modifiée
    pTemp:=PoseMult(pVision, pTemp);
    pPriseVision.trans:=pTemp.trans;
    pPriseVision.rot:=pTemp.rot;
    ! Mouvements du robot par rapport à l'objet de vision
    MoveL pPriseVision, v100, fine, tool1, wobj1;
    ...
ELSE
    ! Gestion d'erreur
    ...
ENDIF
```

5.3 Programmes d'exemple fournis avec la librairie de communication

Vue d'ensemble

La librairie de communication est fournie avec 2 fichiers d'exemple :

- Un exemple de programme robot « TESTVISION.MOD »,
- Un exemple de programme vision « TESTVISION.JOB ».

Ces 2 fichiers d'exemples sont conçus pour fonctionner ensemble pour une mise en œuvre rapide de la communication entre le robot et la caméra « In-Sight ».

Exemple de programme robot

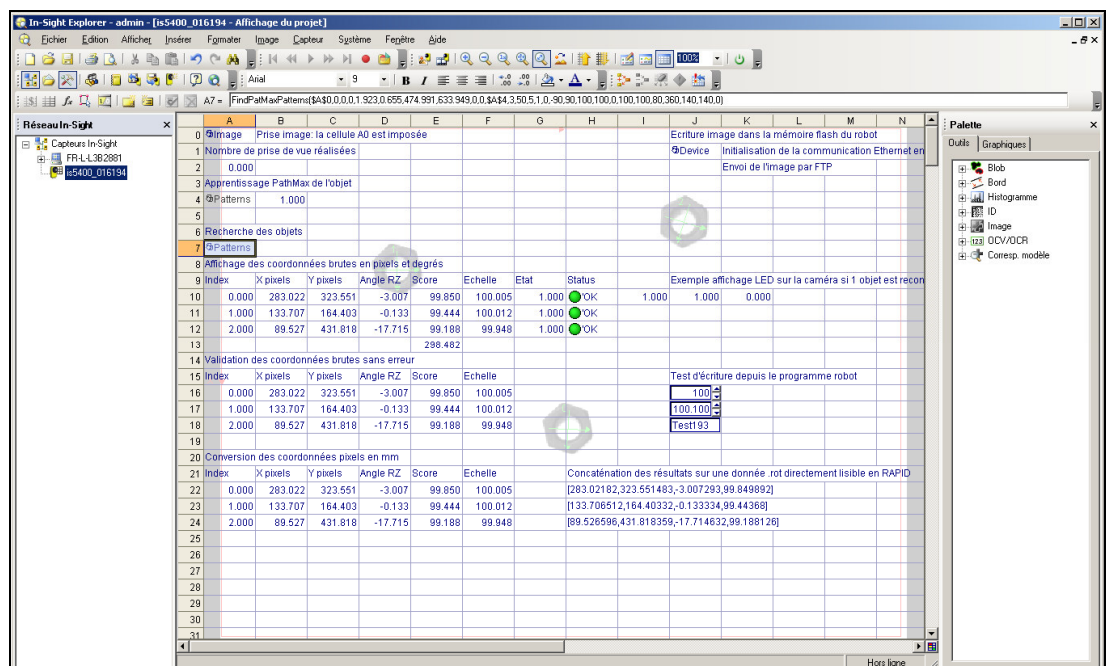
L'exemple de programme robot teste toutes les fonctions de la librairie de communication. Le résultat des échanges de communication est affiché de façon interactive sur le pupitre « FlexPendant » :

- Lancer la routine « TestProduction » pour une exécution sans interaction utilisateur,
- Lancer la routine « TestProduction2 » pour une exécution avec interaction utilisateur.

Exemple de programme vision

L'exemple de programme vision « TESTVISION.JOB » fonctionne avec l'exemple de programme robot, l'ensemble permet de valider les fonctions de la librairie.

Les objets reconnus par la caméra sont 3 écrous à 6 pans de taille « M8 » ou « M10 » ; cet objet a été volontairement choisi pour la simplicité et la robustesse de traitement de vision et parce que l'on peut le trouver facilement partout. La détection de l'objet doit être réalisée sur un fond clair (ex : feuille de papier blanc).



Index	X pixels	Y pixels	Angle RZ	Score	Echelle	Etat	Status
0	0.000	283.022	323.551	-3.007	99.850	100.005	1.000
1	1.000	133.707	164.403	-0.133	99.444	100.012	1.000
2	2.000	89.527	431.818	-17.715	99.188	99.948	1.000

5.4 Configuration de l'adresse IP pour un système de commande IRC5

Vue d'ensemble

La configuration de l'adresse IP pour un système de commande IRC5 est nécessaire pour communiquer par Ethernet avec la caméra « In-Sight ». Cette configuration ne peut être réalisée qu'avec le pupitre « FlexPendant » de l'IRC5 (cette opération ne peut pas être réalisée sur le « Robot Virtuel »).

La consultation de l'adresse IP peut être réalisée à tout moment de deux façons :

- Soit avec le pupitre « FlexPendant »,
- Soit avec le logiciel pour PC « RobotStudio ^{Online} » livré avec le système de commande IRC5.

Port Ethernet à utiliser et configuration par défaut de l'adresse IP

Le système de commande IRC5 est livré en série avec 2 ports Ethernet utilisateur :

- Le port « Service » dédiée aux opérations de maintenance, d'installation et de programmation,
- Le port « LAN » (réseau de l'entreprise).



Prudence :

- Seul le port « LAN » doit être utilisé avec un système de vision (ou tout autre ordinateur externe), car le port « Service » met en oeuvre un serveur « DHCP » dans le système de commande, qui attribue automatiquement des adresses IP. La connexion du port « Service » sur le réseau d'entreprise risque de le rendre complètement inopérant !



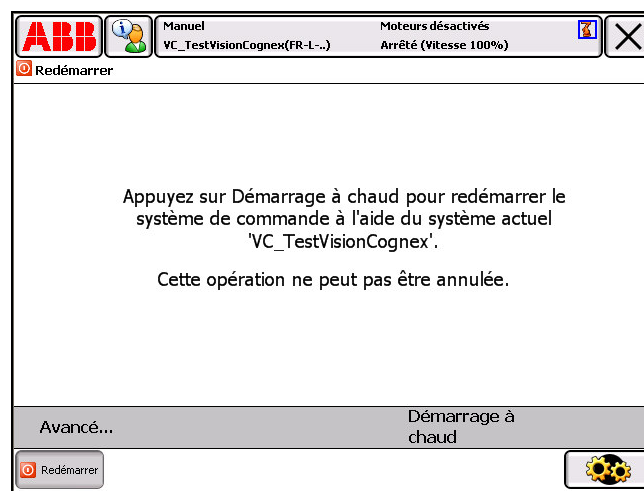
Remarque :

- Le système de commande IRC5 n'a pas d'adresse IP définie par défaut pour le port « LAN », il faut impérativement définir une adresse IP pour pouvoir communiquer avec la caméra.

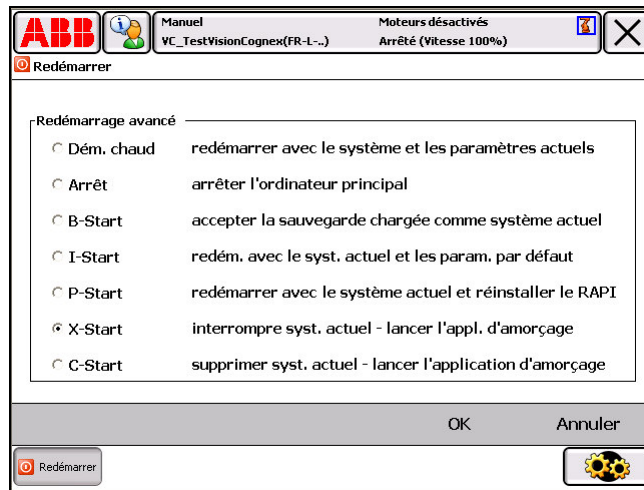
Configuration de l'adresse IP avec le pupitre « FlexPendant »

Procédure :

- Sélectionner le menu principal « ABB », puis le menu « Redémarrer »,
- La fenêtre « Redémarrer » s'ouvre. Choisir le menu « Avancé »,



- La fenêtre « Redémarrage avancé » s'ouvre. Choisir le menu « X-Start »,



La commande « X-Start » lance l'application de chargement du système de commande IRC5 (« ABB Robotics Boot Application » en Anglais). Cette fenêtre contient 4 boutons :

- « Install System »,
- « Settings »,
- « Select System »,
- « Restart Controller ».

Appuyer sur le bouton « Settings » pour accéder à la configuration de l'adresse IP. La fenêtre « Network Connection » s'affiche, le bouton radio « Use no IP address » est activé par défaut.

- Sélectionner le bouton radio « Use the following IP settings » pour configurer votre adresse IP avec le pavé numérique affiché sur la gauche.
- Exemple de configuration d'adresses :
 - IP address : 10.33.48.250
 - Subnet mask : 255.255.255.0
 - Default gateway : 10.33.48.255

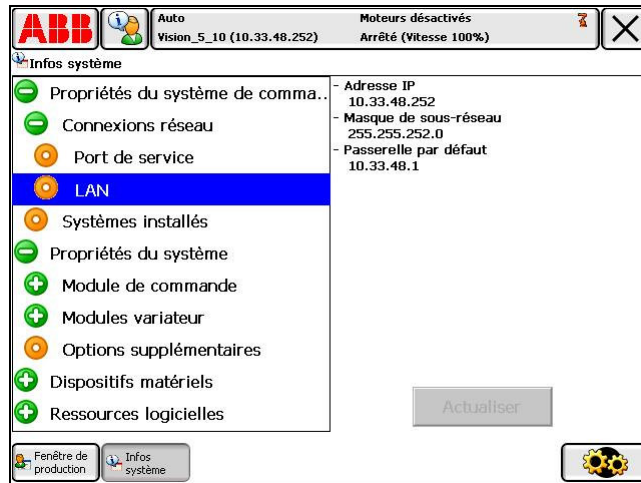
Appuyer sur le bouton « Select system » pour sélectionner un système installé dans la mémoire flash.

Appuyer sur le bouton « Restart Controller » pour redémarrer le système de commande.

Consultation des adresses IP avec le pupitre « FlexPendant »

Procédure :

- Sélectionner le menu principal « ABB », puis le menu « Infos Système ».
- La fenêtre « Infos Système » s'ouvre. Sélectionner « Propriétés du système de commande », puis « connexions réseau » pour visualiser les adresses IP en cours d'utilisation.



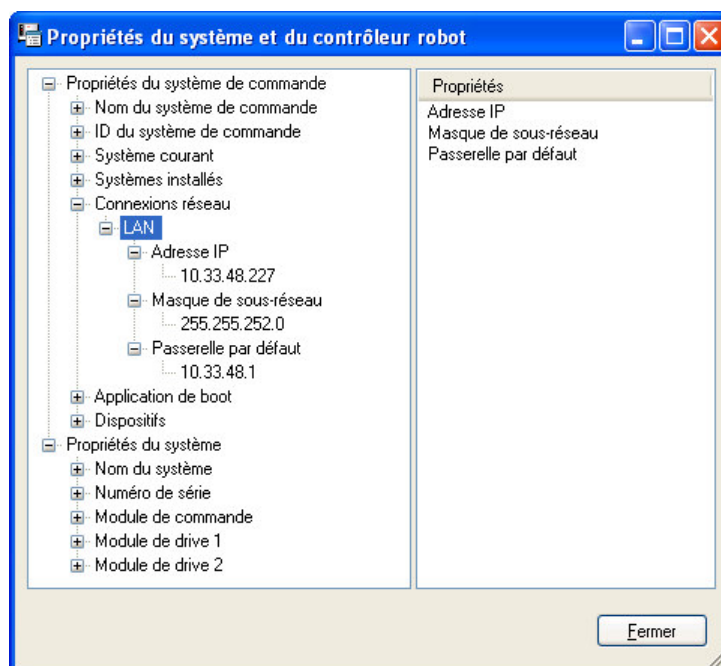
Remarque :

- Depuis la version RobotWare 5.08, l'adresse IP principale est visible en permanence sur le bandeau supérieur d'état du système. Dans l'exemple ci-dessus :
 - Nom du système : « Vision_5_10 »
 - Adresse IP : 10.33.48.252.
- Cette information n'est pas disponible quand on utilise le robot virtuel (uniquement sur une baie de commande réelle).

Consultation des adresses IP avec le logiciel PC « RobotStudio ^{Online} »

Procédure :

- Sélectionner dans l'arborescence « Vue du robot », l'élément « Systèmes de commande », puis dans son menu contextuel (clic droit) « Ajouter un système de commande ».
- Sélectionner votre système de commande dans la liste proposée, celui-ci est alors ajouté dans l'arborescence « Vue du robot ». Son adresse IP principale est affichée à droite du nom du système.
- Pour accéder à l'ensemble de la définition de l'adresse IP, sélectionner dans le menu contextuel « Propriétés », puis « Propriétés du système et du contrôleur robot », qui affiche la boîte de dialogue suivante :



Remarque :

Cette procédure n'est pas valable pour un « Robot Virtuel » qui n'a pas d'adresse IP propre, car il utilise celle du PC qui héberge le logiciel « Virtual IRC5 » ou « RobotStudio 5 ».

5.5 Configuration de l'adresse IP pour une caméra « In-Sight »

Vue d'ensemble

La configuration de l'adresse IP pour une caméra « In-Sight » est réalisée avec l'utilitaire « In-Sight Connection Manager » de la société COGNEX (se reporter au manuel d'utilisation correspondant).



Remarque :

L'adresse IP choisie doit être compatible avec celle du robot. Dans le cas d'installations réseaux de taille limitée, seul le dernier octet de l'adresse IP change ; c'est ce que propose le logiciel « In-Sight Connection Manager » par défaut.

6 Interface Homme-Machine “IHM”

Vue d'ensemble

La librairie de communication ne met pas en œuvre d'interface homme machine spécifique à la vision sur le pupitre « FlexPendant ». Les messages d'avertissements et d'erreurs utilisent l'interface homme machine standard du pupitre « FlexPendant ». Il y a deux types de messages d'erreur :

- Les messages d'erreur générés par la bibliothèque de communication ABB,
- Les messages d'erreur générés par la caméra « In-Sight ».

6.1 Messages d'erreurs générés par la bibliothèque de communication ABB

Dans cette version, les messages d'erreurs générés par la bibliothèque de communication sont définis selon les 3 rubriques ci-dessous.

Erreurs de communication « socket messaging »

Code	Description	Routine
-100	Initialisation impossible.	CX_InitComm
-101	Ecriture de données impossible.	Toutes les routines
-102	Lecture de données impossible.	Toutes les routines
-103	Le délai de réception de données a expiré.	Toutes les routines

Erreurs d'application vision générales

Code	Description	Routine
-200	Numéro de caméra incorrect (hors de la plage « 1 » à « 4 »).	Toutes les routines avec paramètre « CameraId »
-201	Format incorrect des coordonnées vision de la cellule du programme vision (format ABB).	CX_GetVisionData
-202	Coordonnées de vision hors tolérance. La valeur est en dehors des tolérances minimum et maximum définies dans l'instruction « CX_Setup_Camera ».	
-203	Score de vision hors tolérance. La valeur est inférieure à la tolérance définie dans l'instruction « CX_Setup_Camera ».	
-204	Fonction de communication non implémentée dans cette version de la librairie.	CX_GetInfo

Erreurs d'application vision concernant l'étalonnage robot-vision

Code	Description	Routine
-250	Nombre incorrect de positions d'étalonnage.	SetWObjVision
-251	Nombre insuffisant de positions d'étalonnage robot.	SetWObjVision
-252	Nombre insuffisant de positions d'étalonnage vision.	SetWObjVision
-253	Le repère objet vision n'a pas pu être calculé, les positions d'étalonnage sont incorrectes ou incohérentes.	SetWObjVision

Erreurs d'application vision spécifiques à la caméra « In-Sight »

Code	Description	Routine
-300	Code renvoyé par la caméra « In-Sight » inconnu.	Toutes les routines
-301	Identifiant incorrect de ligne de la cellule du programme vision (hors 0 ... 399).	CX_GetValue CX_GetVisionData CX_SetFloat
-302	Identifiant incorrect de colonne de la cellule du programme vision (hors A ... Z).	CX_SetInteger CX_SetString

6.2 Messages d'erreurs générés par la caméra « In-Sight »

Dans cette version, les messages d'erreurs générés par la caméra « In-Sight » sont les suivants :

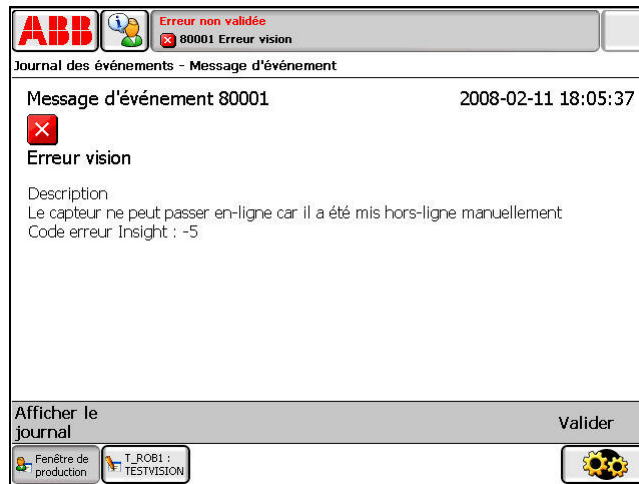
Code	Description	Routine
0	La commande est inconnue.	Toutes les routines
-1	Le nombre est hors limite (0 à 8), ou n'est pas un entier.	CX_SendMessage CX_SetEvent CX_SetEventWait CX_SetOnline CX_TriggerImage
	L'identifiant de la cellule est invalide.	CX_GetValue
	L'identifiant de la cellule est invalide ou la valeur n'est pas un entier.	CX_SetInteger
	L'identifiant de la cellule est invalide ou la valeur n'est pas un réel.	CX_SetFloat
	Le nom du job est manquant.	CX_LoadFile
-2	La commande n'a pas pu être exécutée, ou le système de vision est hors-ligne.	CX_GetInfo CX_GetValue CX_SendMessage CX_SetEvent CX_SetEventWait CX_SetFloat CX_SetInteger CX_SetOnline CX_SetString CX_TriggerImage
	Le job actif n'a pas été sauvegardé, la commande n'a pas pu être exécutée.	CX_GetFile
	Le job n'est pas chargé, système de vision en ligne ou fichier introuvable.	CX_LoadFile
-4	Le job n'est pas chargé, le capteur manque de mémoire.	CX_LoadFile
-5	Le capteur ne peut passer en ligne car il a été mis hors-ligne manuellement.	CX_SetOnline
-6	L'utilisateur n'a pas les droits requis pour exécuter cette commande.	CX_LoadFile CX_SendMessage CX_SetEvent CX_SetEventWait CX_SetFloat CX_SetInteger CX_SetOnline CX_SetString CX_TriggerImage

6.3 Message d'information généré par les fonctions de la librairie de communication

**Note :**

Les codes d'erreur de la caméra « In-Sight » numéro **-1** et **-2** prennent une signification différente en fonction des routines utilisées.

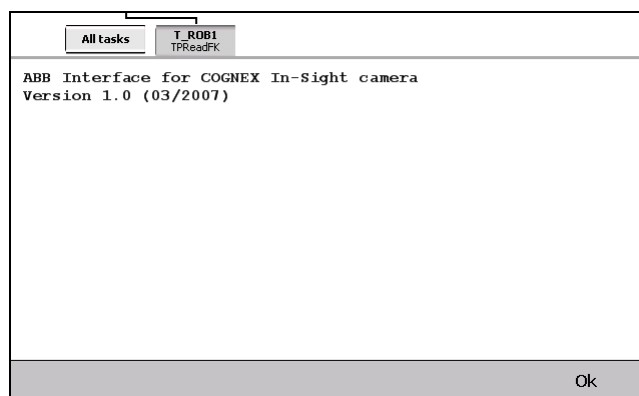
Exemple d'affichage en cas de caméra qui est resté en mode « hors-ligne » (IHM configurée en Français):



6.3 Message d'information généré par les fonctions de la librairie de communication

Seule l'instruction « CS_ShowVersion » génère un affichage d'information sur le pupitre « FlexPendant » dans la fenêtre opérateur (voir le paragraphe 4.1.9).

Résultat de cet affichage sur le pupitre « FlexPendant » :



6.4 Affichage d'image de caméra « In-Sight » sur le pupitre « FlexPendant »

La principale utilisation de cette fonction est l'affichage de l'image d'objets défectueux pour que l'opérateur puisse suivre facilement les aléas éventuels de production et prendre les mesures appropriées.



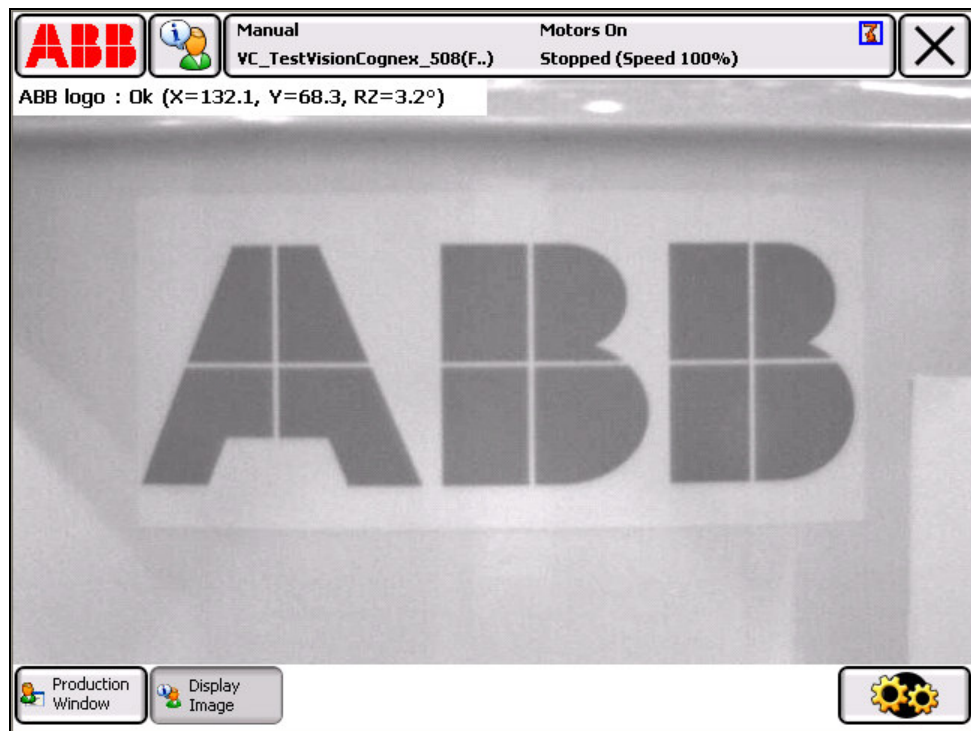
Remarque :

Cette fonction n'est pas disponible comme produit, elle est mentionnée comme un exemple de ce qui est techniquement possible.

Le principe de mise en œuvre de cette fonction est le suivant :

- Le programme robot demande une prise de vue,
- Le programme vision détecte une anomalie de l'objet et mémorise l'erreur,
- Le programme vision envoie l'image au système de commande IRC5 avec l'instruction « WriteImageFTP » (voir le paragraphe 8.3),
- Le programme robot détecte l'anomalie lors de la demande de lecture des coordonnées,
- Le programme robot demande l'affichage de l'image envoyé par le programme vision sur le pupitre « FlexPendant ».

Exemple d'affichage préliminaire réalisé pour une démonstration lors d'un salon :



7 Procédures d'étalonnage robot – vision

7.1 Mise en œuvre d'un système de vision pour le guidage d'un robot

Etalonnage du plan image

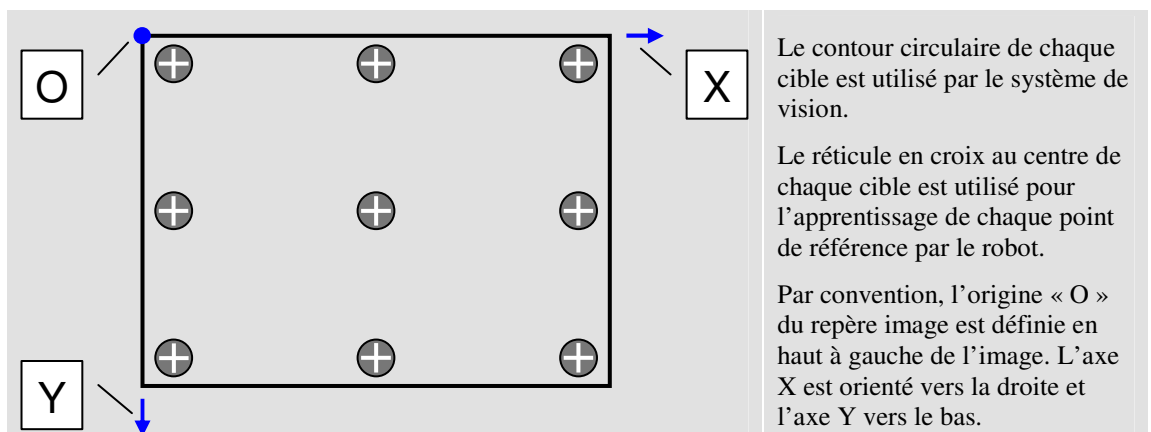
L'utilisation d'un système de vision pour réaliser une opération de guidage de robot nécessite l'obtention de coordonnées de positionnement d'un objet exprimées en unité métrique :

- La position native d'un objet est toujours mesurée en unité de pixels dans le repère capteur.
- Une opération de « mise à l'échelle » dans le système métrique est nécessaire pour obtenir des coordonnées cartésiennes exprimées en mm utilisables par le système de commande du robot. Cette opération est en fait un **étalonnage du plan image**, et dépend de plusieurs facteurs :
 - La géométrie du capteur de la caméra (nombre et forme des pixels...),
 - L'angle de champ de l'objectif (défini par sa focale pour une taille de capteur donnée),
 - La distance de focalisation de l'objectif,
 - La correction éventuelle de distorsions optiques de l'objectif.
- Cette opération d'étalonnage du plan image doit impérativement être réalisée **avant** l'opération d'étalonnage robot – vision.
- La présente documentation ne traite pas de la réalisation de cet étalonnage du plan image, veuillez vous reporter à la documentation COGNEX du logiciel « In-Sight Explorer » :
 - Pour « In-Sight Explorer » version 3.2 et inférieur : se référer aux fonctions « Calibrate » (qui met en œuvre 4 points de référence), et « CalibrateAdvanced » (qui met en œuvre jusqu'à 9 points de référence).
 - Pour « In-Sight Explorer » version 4.0 et supérieur : se référer au nouvel assistant « EasyBuilder », qui inclut une fonction d'étalonnage automatique.

Dans tous les cas, cette opération d'étalonnage du plan image nécessite la mise en œuvre d'une **mire d'étalonnage** qui sera traitée par le système de vision :

- La taille de la mire doit être adaptée à la surface du plan image,
- La mire est composée de plusieurs cibles qui sont reconnues par le système de vision. Ces cibles sont très souvent circulaires et leur centre matérialise un point de référence,
- La position de chaque cible sur la mire doit être connue (ou mesurée) dans un repère défini sur le plan image,
- L'algorithme d'étalonnage du plan image utilise une paire de coordonnées pour chaque cible pour calculer les paramètres d'étalonnage :
 - 2 coordonnées (X, Y) caméra en pixels,
 - 2 coordonnées (X, Y) dans le plan image en mm.
- Dans la pratique, cette mire doit pouvoir aussi être utilisée avec le robot par apprentissage.

Exemple de mire d'étalonnage à 9 cibles avec affichage du repère (O, X, Y) du plan image



7.2 Etalonnage robot – vision dans le cas d'une caméra fixe

Vue d'ensemble

La procédure d'étalonnage robot – vision est indispensable pour l'obtention de mouvements précis du robot en fonction des coordonnées envoyées par le système de vision. Cette procédure permet de définir avec précision la position et l'orientation du repère caméra **par rapport au repère de base du robot**.

Matériel et repères géométriques mis en œuvre dans le cas d'une caméra fixe

Dans le cas d'une caméra fixe, la procédure nécessite la mise en œuvre d'une **mire d'étalonnage** qui sera apprise à la fois par le robot et par le système de vision. La mire d'étalonnage est placée dans le champ observé par caméra à la distance nominale de mise au point de l'objectif. La mire est constituée de plusieurs références géométriques dont les coordonnées sont connues (voir la définition détaillée au paragraphe suivant).

Le matériel et les repères mis en œuvre par la procédure d'étalonnage robot/vision sont illustrés par la figure ci-dessous :

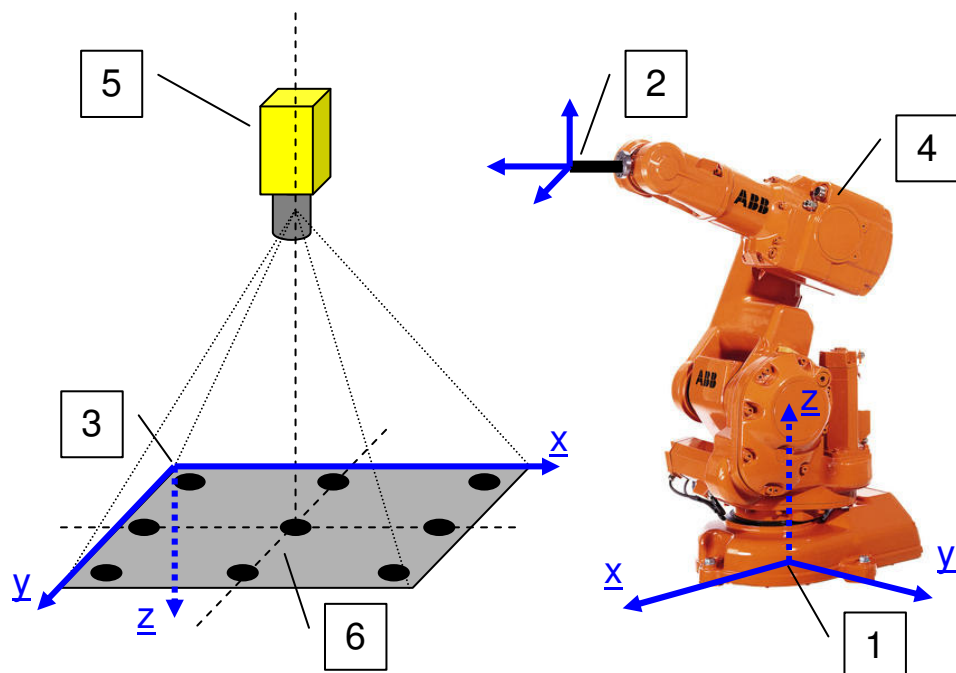


Figure 3. Repères géométriques mis en œuvre dans le cas d'une caméra fixe

1	Repère de base du robot
2	Repère outil du robot
3	Repère objet du robot (ce repère doit coïncider avec le repère caméra)
4	Robot
5	Caméra « In-Sight » série « 5000 » avec objectif
6	Mire d'étalonnage avec références géométriques multiples (9 cibles dans cet exemple) située dans le plan image des objets à traiter

Mise en œuvre du calcul d'étalonnage robot – vision dans le cas d'une caméra fixe

Le but de l'étalonnage robot – vision dans le cas d'une caméra fixe est de calculer un **repère objet** qui coïncide avec le repère caméra, de telle sorte que les coordonnées 2D (X, Y, RZ) fournies par le système de vision soient directement utilisables comme déplacement dans le repère objet.

L'étalonnage robot – vision nécessite la mise en œuvre de la **mire d'étalonnage** décrite dans le paragraphe précédent. Deux méthodes sont possibles :

- Définition du repère objet directement sur la mire d'étalonnage par apprentissage de 3 points, à condition que l'origine et les axes X et Y soient matérialisés sur la mire :
 - Le point 1 définit l'origine du repère objet,
 - Le point 2 définit l'axe X du repère objet,
 - Le point 3 définit l'axe Y du repère objet.
- Définition du repère objet par plusieurs paires de coordonnées pour chaque cible (3 mini) :
 - 3 coordonnées (X, Y, Z) en mm de la position de la cible dans le repère de base du robot,
 - 2 coordonnées (X, Y) en mm de position de la cible dans le repère du plan image vision.



Remarque:

La deuxième méthode est recommandée: plus facile à implémenter en pratique et plus précise.

Exemple de routine d'apprentissage des positions d'étalonnage dans le cas d'une caméra fixe

Exemple avec paires de coordonnées, fournis dans le programme robot « TESTVISION.MOD ».

```
! Nombre de positions d'étalonnage
CONST num nbCALPOS:=9;
! Déclaration des positions d'étalonnage en tableau
CONST robtargt pEtalonRobot{nbCALPOS}:=[ ... ];
! Déclaration des transformations de mesure vision en tableau
PERS pose pEtalonVision{nbCALPOS}:=[ ... ];
VAR num nScore;
VAR num nError;
VAR bool bOk;
...
PROC Etalonnage()
    ! Mouvement sur les positions d'étalonnage
    MoveL pCalRobot{1},v20,fine,tool1;
    Stop;
    MoveL pCalRobot{2},v20,fine,tool1;
    Stop;
    ...
    MoveL pCalRobot{9},v20,fine,tool1;
    Stop;
    ...
    ! Lecture des coordonnées vision
    bOk:=CX_GetVisionData(1,"F",21,pCalVision{1},nScore,nError);
    bOk:=CX_GetVisionData(1,"F",22,pCalVision{2},nScore,nError);
    ...
    bOk:=CX_GetVisionData(1,"F",29,pCalVision{9},nScore,nError);
    !
```

```
! Calcul de l'étalonnage robot-vision
SetWobjVision pCalRobot,pCalVision,nbCALPOS,wobjVision,nError;
ENDPROC
```



Remarque :

L'instruction « SetWobjVision » est fournie par le module « INSIGHT.SYS ».



Conseil :

L'intérêt de séparer l'étalonnage du plan image de l'étalonnage robot – vision est de faciliter la maintenance de l'installation en cas d'intervention sur la caméra :

- La caméra et/ou l'objectif peuvent être changés sans avoir à refaire l'étalonnage robot – vision,
- Il suffit de repositionner la mire d'étalonnage avec la précision nécessaire et de relancer le traitement de vision.

7.3 Etalonnage robot – vision dans le cas d'une caméra embarquée sur le robot

Vue d'ensemble

Le but de la procédure d'étalonnage robot – vision en cas de caméra embarquée sur le robot est identique à la procédure décrite dans le paragraphe précédent, mais sa mise en œuvre est différente, en particulier quant au matériel d'étalonnage et aux repères utilisés. Cette procédure permet de définir avec précision la position et l'orientation du repère caméra **par rapport au repère de la bride du robot** (« tool0 »).

Matériel et repères géométriques mis en œuvre

Dans le cas d'une caméra embarquée sur le robot, la mire mise en œuvre précédemment sera remplacée par une référence sphérique pour permettre une large plage de mouvements de rotation de la caméra autour du centre de la sphère.

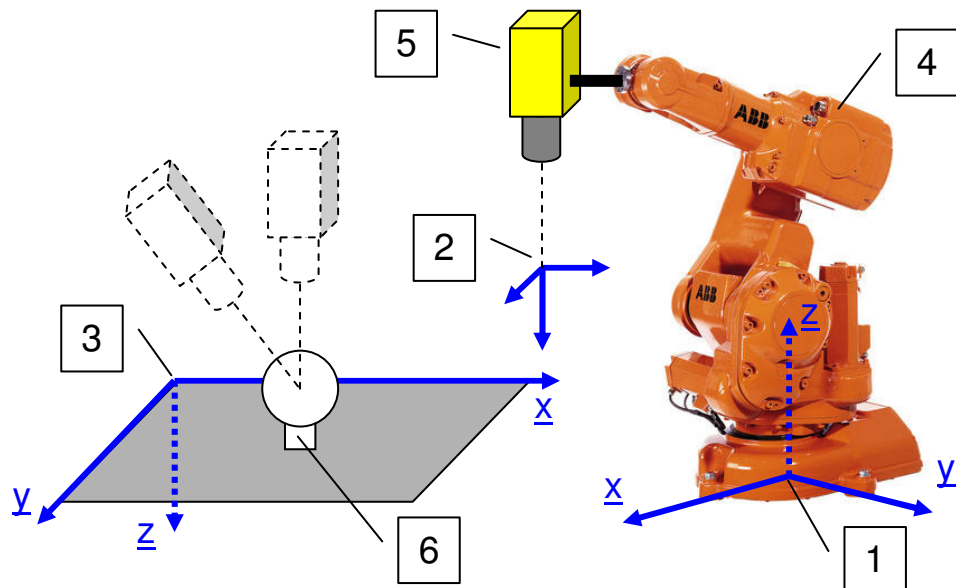


Figure 4. *Repères géométriques mis en œuvre dans le cas d'une caméra embarquée sur le robot*

1	Repère de base du robot
2	Repère outil du robot (ce repère doit coïncider avec le repère caméra)
3	Repère objet du robot
4	Robot
5	Caméra « In-Sight » série « 5000 » avec objectif
6	Mire d'étalonnage avec référence sphérique

Mise en œuvre du calcul d'étalonnage dans le cas d'une caméra embarquée sur le robot

Le but de l'étalonnage robot – vision dans le cas d'une caméra embarquée sur le robot est de calculer un **repère outil** aligné avec le capteur de la caméra. Deux méthodes sont possibles :

- Une méthode simplifiée qui nécessite une mire d'étalonnage avec une référence sphérique, qui est similaire à la procédure d'étalonnage standard d'un repère outil fournie avec le système de commande IRC5,
- Une méthode avancée qui nécessite une mire d'étalonnage et des algorithmes particuliers.

La mise en œuvre de cette procédure est plus complexe que dans le cas d'une caméra fixe. Veuillez contacter ABB France division Robotique pour un éventuel support technique.

8 Procédures particulières

8.1 Utilisation du « Robot Virtuel » pour tester une application de vision

Vue d'ensemble

Le « Robot Virtuel » permet de tester sur un ordinateur compatible PC une application de communication entre un système de commande IRC5 et une caméra « In-Sight », donc sans avoir à disposer matériellement d'un robot.

Ce type de test hors-ligne est particulièrement intéressant pour l'intégrateur qui doit préparer un nouveau projet pour un client, ou un utilisateur final qui veut tester une modification sans arrêter sa production.

Mise en œuvre du « Robot Virtuel »

Le « Robot Virtuel » pour le système de commande IRC5 est disponible avec les produits logiciels:

- « Virtual IRC5 », ou
- « RobotStudio 5.0 »

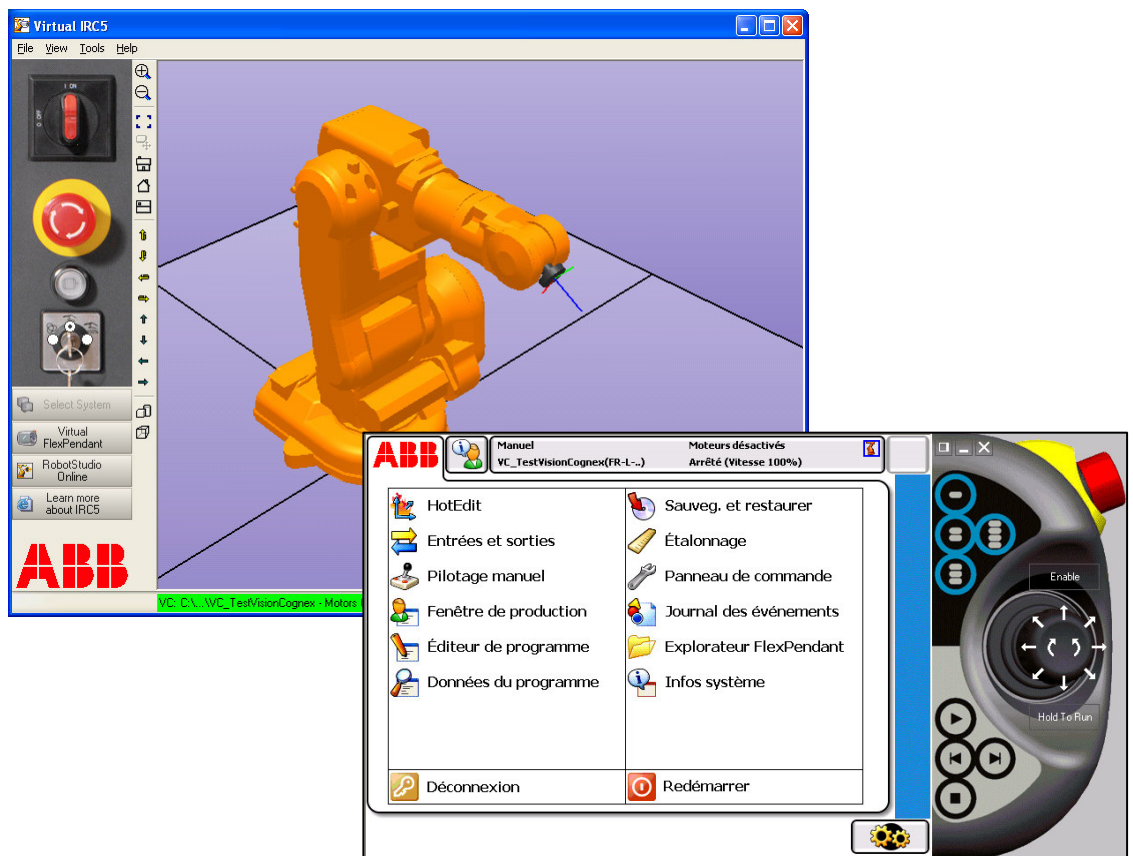


Figure 5. Logiciel « Virtual IRC5 »



Conseil :

Vous pouvez obtenir plus de renseignements sur le « Robot Virtuel » et les produits logiciels ABB avec le site Internet suivant : <http://www.abb.com/roboticssoftware>.

Séquence des opérations à réaliser pour mettre en œuvre le « Robot Virtuel » :

- Connecter le câble Ethernet de la caméra « In-Sight » sur l'ordinateur compatible PC,
- Connecter le câble d'alimentation de la caméra « In-Sight » sur le module d'extension relié à une alimentation 24 V,
- Configurer avec le logiciel « RobotStudio Online » une instance du « Robot Virtuel » en fonction de la configuration du système de commande IRC5 qui doit être simulé,
- Configurer l'adresse IP de la caméra « In-Sight » (l'adresse doit être compatible avec celle du PC) avec le logiciel « In-Sight Connection Manager » de la société COGNEX,
- Démarrer le « Robot Virtuel » depuis les logiciels « Virtual IRC5 » ou « RobotStudio 5.0 »,
- Charger et lancer le programme de communication robot/vision dans le « Robot Virtuel ».

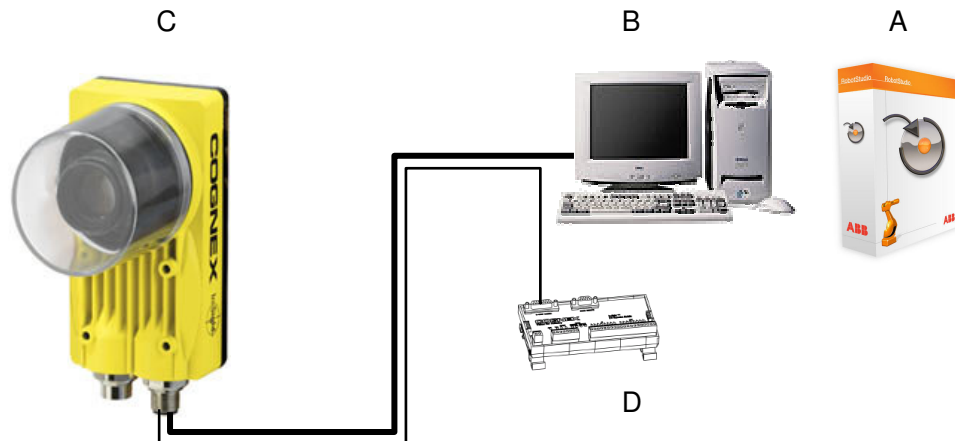


Figure 6. Architecture matérielle avec un PC et le « Robot Virtuel »

A	Robot Virtuel – logiciels « Virtual IRC5 » ou « RobotStudio 5.0 »
B	Ordinateur compatible PC
C	Caméra « In-Sight » série « 5000 »
D	Module d'extension de la caméra « In-Sight » pour son alimentation 24 V

Limites d'utilisation

Le « Robot Virtuel » exécute l'instance du système de commande IRC5 à simuler dans un répertoire défini par l'utilisateur. Du fait que ce répertoire ne peut pas être la racine du disque d'installation du « Robot Virtuel », certaines opérations sur les fichiers réalisées sur le robot réel ne pourront pas être exécutées telles quelles par le « Robot Virtuel » sans une adaptation.



Remarque :

Dans le cas d'une application de communication entre un système de commande IRC5 et une caméra « In-Sight », un exemple type sera l'écriture d'une image par la caméra dans la mémoire flash du système de commande IRC5 avec la fonction « WriteImageFTP » : dans ce cas, il faudra utiliser un répertoire de destination modifié pour l'écriture du fichier, en fonction du répertoire choisi par utilisateur pour l'instance du système de commande IRC5 à simuler.

8.2 Mode de mise au point « debug »

Vue d'ensemble

Le mode « debug » facilite la mise au point d'une application de vision en affichant au pupitre toutes les trames de communication échangées en le robot et la caméra « In-Sight », dans la fenêtre de dialogue opérateur au format texte.

Mise en œuvre

Le mode « debug » est mis en œuvre avec l'instruction « CX_SetDebugMode » et un des 2 paramètres mutuellement exclusifs « On » | « Off » (voir la description de référence de l'instruction au chapitre 4.1.7).

Affichage dans la fenêtre « dialogue opérateur »

Exemple d'affichage sur le pupitre « FlexPendant » en mode « debug » (IHM configurée en Anglais) :

- Ouverture de la session de communication avec une caméra « in-Sight », avec nom d'utilisateur (« admin » par défaut) et mot de passe (vide par défaut),
- Commande de lecture du nom du programme vision en cours (« GF » pour « GetFile »), la caméra retourne « 1 » (pour un acquittement positif), puis le nom du programme (« TestCameraInsight.job »).

All tasks	
T_ROB1->Socket start client done...	
T_ROB1->Com. vision : Welcome to In-Sight(tm) 5400 Session 0	
User:	
T_ROB1->Com. robot : admin	
T_ROB1->Telnet user name done...	
T_ROB1->Com. vision : Password:	
T_ROB1->Com. robot :	
T_ROB1->Telnet password done...	
T_ROB1->Com. vision : User Logged In	
T_ROB1->Com. robot : GF	
T_ROB1->Com. vision : 1	
TestCameraInsight.job	
Effacer	Ne pas afficher les journaux

8.3 Transfert d'image de la caméra « In-Sight » vers le système de commande IRC5

Vue d'ensemble

Le transfert d'images de la caméra « In-Sight » vers le système de commande IRC5 est réalisé avec le protocole de communication « FTP ». Cette possibilité est notamment utilisée pour visualiser et mémoriser l'image d'objets défectueux sur un ordinateur connecté au réseau Ethernet.



Note :

L'affichage d'une image sur le pupitre de programmation « FlexPendant » sera disponible comme un module optionnel dans une version ultérieure de la librairie de communication. Veuillez consulter ABB France Division Robotique pour plus de renseignement.

Programmation vision

Le transfert d'une image de la caméra vers le système de commande IRC5 est réalisé avec l'instruction « WriteImageFTP » (se référer au logiciel « In-Sight Explorer »). Liste des paramètres à configurer :

Nom de l'hôte	Utilisez l'adresse IP du système de commande IRC5 pour le port Ethernet « LAN ».
Nom d'utilisateur	Par défaut : aucun utilisateur n'est configuré pour le système de commande IRC5.
Mot de passe	Par défaut : non utilisé.
Structure image	Par défaut : \$A\$0 (coordonnée de la cellule d'acquisition de l'image dans le programme vision).
Nom du fichier	L'image est mémorisée dans la mémoire « flash » du système de commande IRC5 : unité de disque « hd0a ». Ajoutez le nom du répertoire et du fichier image.
Valeur max	Utilisez « 0 » pour écrire toujours dans la même image, sinon limitez le nombre d'images pour éviter de saturer la mémoire « flash ».
Remise à zéro	Utilisez toujours cette option pour éviter d'écrire un nombre illimité d'images.
Format des données	Utilisez le format « JPEG » (choix : JPG) pour optimiser la taille mémoire utilisée.

9 Evolutions logicielles

Vue d'ensemble

Ce chapitre liste toutes les évolutions logicielles réalisées sur le produit « Interface vision pour COGNEX In-Sight pour système de commande IRC5 ».

Version 1.0.0 (avril 2007)

Nouvelles fonctions :

- Première version disponible comme produit officiel.
 - Ajout de l'interface en Hollandais (en plus du Français et de l'Anglais).
-

Version 1.0.1 (mai 2007)

Nouvelles fonctions :

- Ajout de l'interface en Allemand.

Corrections :

- Correction d'une erreur d'index qui se traduisait par l'affichage d'un message erroné en cas d'erreur de numéro de caméra (numéro en dehors de la plage 1 à 4).
-

Version 1.0.2 (février 2008)

Nouvelles fonctions :

- Ajout de l'interface en Espagnol,
- Ajout de l'interface en Italien,
- Ajout de l'instruction « CX_SetFile »,
- Ajout de l'instruction « SetWObjVision ».

Corrections :

- Correction d'une erreur sur l'affichage des erreurs numéro -103 et -300.

Mise à jour de la documentation.

Version 1.0.3 (avril 2009)

Nouvelles fonctions :

- Ajout de l'interface en Suédois,
- Possibilité de ne charger que certains modules de location de l'IHM (au lieu de charger tous les modules dans les versions précédentes)

Documentation :

- Correction de la description de l'instruction "SetToolVision".
- Mise à jour en fonction des changements ci-dessus,
- Mise à jour mineure du format,
- Ajout références à la nouvelle version 4 du logiciel « In-Sight Explorer » avec l'assistant « EasyBuilder ».

10 Index et tables

10.1 Index

Section non utilisée dans cette version de documentation.

10.2 Table des illustrations

<i>Figure 1.</i>	<i>Architecture matérielle avec un système de commande IRC5</i>	<i>11</i>
<i>Figure 2.</i>	<i>Chaînage de repères géométriques utilisé pendant l'exploitation.....</i>	<i>61</i>
<i>Figure 3.</i>	<i>Repères géométriques mis en œuvre dans le cas d'une caméra fixe.....</i>	<i>75</i>
<i>Figure 4.</i>	<i>Repères géométriques mis en œuvre dans le cas d'une caméra embarquée sur le robot</i>	<i>79</i>
<i>Figure 5.</i>	<i>Logiciel « Virtual IRC5 »</i>	<i>80</i>
<i>Figure 6.</i>	<i>Architecture matérielle avec un PC et le « Robot Virtuel ».....</i>	<i>81</i>