

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique



Ecole Nationale Polytechnique
Département Automatique



Mémoire de fin d'études en vue de l'obtention du diplôme d'Ingénieur d'Etat en Automatique

Thème:

Gestion de la station CE117 par l'automate S7 314 IFM

Houssem Eddine ZERARKA
Moufek KHELIFI

Présenté devant le jury composé de :

Président:	M.TADJINE	Professeur à l'ENP
Rapporteur:	E.M.BERKOUK	Professeur à l'ENP
Examineur:	O.STIHI	Chargé de Cours à l'ENP

JUIN 2016

ENP 10, Avenue Hassen Badi, BP. 182, 16200 El Harrach, Alger, Algérie

ملخص:

العنوان: تسيير المحطة CE117 باستعمال المبرمج S7 314 IFM
العمل المنجز في المذكرة يتمحور على استعمال مسير صناعي مبرمج "سيمنس" من خلال برنامج STEP7 و WinCC
لأجل ذلك سمحت لنا دائرة الأوتوماتيك للمدرسة العليا المتعددة التقنيات باستعمال محطة تسيير وضبط الأنظمة الصناعية
(مستوى الماء، التدفق، الضغط، درجة الحرارة) TecQuipmentProcess Trainer CE117 .
باستعمال مبرمج سيمنس S7 314 IFM قمنا بالتمنجه والتحكم بطرق مختلفة في جميع أنظمة المحطة .
للتمنجه وتركيب المنظم، استعملنا طريقة برويدا ، طريقة موازنة الأقطاب ومجموعة طرق أخرى.

الكلمات المفتاحية : مسير صناعي مبرمج "سيمنس", "برنامج" ستابسات , "برنامج" وينسيسيفلكسييل , نموذج محطة ضخ المياه
TecQuipmentProcess Trainer CE117

Abstract:

Subject: management of the station CE117 Process Trainer with the PLC Siemens S7 300.

The work presented in this paper is based primarily on the use of programmable SIEMENS via the software STEP7 and WinCC.

For that, The Department of Automation of the national polytechnic school let us to use the station CE 117 Process Trainer.

Using the programmable logic controller S7 314 IFM, we identified and regulated by a different method all sub-system of the station CE117.

For the identification and determination of regulators applied, we used the method of Broida, the methods to compensate for the poles (P, PI), sliding mode, adaptive control and adaptive fuzzy control.

Key words: CE117 Process Trainer, Siemens PLC S7-313C, STEP7, WinCC.

Résumé:

Le travail présenté dans ce mémoire est basé essentiellement sur l'utilisation des automates programmables SIEMENS par le biais des logiciels STEP 7 et WinCC. Pour ce faire, le Département d'Automatique de l'Ecole Nationale Polytechnique nous a permis d'utiliser une station de contrôle des processus (débit, niveau, pression et température), TecQuipmentProcess Trainer CE117.

À l'aide de l'automate S7 IFM 314, nous avons identifié et régulé par des différentes méthodes l'ensemble de sous-systèmes constituant cette installation.

Pour l'identification, et la synthèse de régulateur, nous avons utilisé la méthode de broida, les méthodes de compensation des pôles (P, PI), mode de glissement, ainsi que la commande adaptative et l'adaptation par logique floue.

Mots clés : CE117 Process Trainer, Siemens S7-300, Wincc, Step7.

Dédicace

Je dédie ce travail tout d'abord à mon père qui m'a tout appris, tant donné sans rien demandé en retour.

A ma très chère mère sans laquelle je ne serais pas l'homme que je suis.

A mon frère Brahim qui a su m'épauler dans les moments difficiles.

A mes frères Abd-elbasset et Fayçal

A mon ami et binôme de la vie Moufek sur qui j'ai toujours pu compter.

A mes amis et camarades avec qui j'ai passé de bons moments.

A tous ceux qui ont contribué de loin ou de prêt à notre travail.

Zerarka Housseem Eddine

Je dédie ce modeste travail tout d'abord à mon père qui m'a donné tous sans rien demandé en retour.

A ma chère mère et ma grande mère sans elles je ne serais pas l'homme que je suis.

A mes frères Amine et Adel

A ma petite sœur

A mon ami et binôme de la vie Housseem sur qui j'ai toujours pu compter.

A toute ma famille

A mes amis et camarades avec qui j'ai passé de bons moments.

A tous ceux qui ont contribué de loin ou de prêt à notre travail.

KHELIFI Moufek

Remerciements

Nous tenons à remercier dieu de nous avoir donné la force morale, physique et l'aide pour accomplir ce modeste travail.

Nous tenons à remercier notre promoteur Pr. BERKOUK pour nous avoir acceptées encadrées et dirigées durant l'élaboration de ce travail ainsi que pour leur assistance et tous leurs conseils.

Nous remercions chaleureusement les membres du jury pour l'honneur qu'ils nous ont fait en acceptant d'évaluer notre projet.

Nous souhaitons aussi remercier tous les enseignants de l'Ecole Nationale Polytechnique d'Alger, et en particulier, Nos professeurs d'Automatique qui nous ont encadrées auparavant et tous nos enseignants pour les connaissances qu'ils nous ont transmis, leur disponibilité et leurs efforts.

Nous remercions chaleureusement nos familles, pour leurs soutien sans faille, leurs présence émotionnelle ainsi que les nombreux conseils qu'elles nous ont prodigué, et qui nous ont indéniablement permis de mener à bien ce travail.

Que tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail trouvent ici l'expression de notre sincère gratitude.

Sommaire

Liste des tableaux

Liste des figures

Introduction générale	14
Chapitre I : Les automates programmables industriels	
Introduction.....	15
PARTIE A : Description des automates programmables.....	15
I.A.1.Historique	15
I.A.2.Définition	15
I.A.3.Domaine d'application	16
I.A.4.L'architecture	17
a. Module d'alimentation "PS"	18
b. L'unité centrale "CPU"	18
<i>b.1- Processeur.....</i>	<i>19</i>
<i>b.2- Mémoire.....</i>	<i>19</i>
c. Coupleurs "IM"	19
d. Les modules d'entrées/sorties "SM"	19
<i>d.1 Les informations traitées par l'automate</i>	<i>20</i>
<i>d.2 Les caractéristiques des entrées et des sorties d'un automate.....</i>	<i>20</i>
e. Le module de fonction "FM"	20
f. Processeur de communication "CP".....	21
g. Les auxiliaires	21
I.A.5. Langages de programmation	21
a. Les langages graphiques	22
<i>a.1. Le Grafcet.....</i>	<i>22</i>
<i>a.2. Ladder Diagram</i>	<i>22</i>
<i>a.3. Bloc fonction diagramme</i>	<i>23</i>
b. Les langages textuels	24
<i>b.1. Texte Structuré</i>	<i>24</i>
<i>b.2. Liste d'Instructions</i>	<i>25</i>
I.A.6. Traitement du programme par automate.....	26
a- Traitement interne.....	26
b- Lecture des entrées.....	26
c- Exécution du programme	26
d- Ecriture des sorties	26
e- Le temps de réponse total (TRT)	27

I.A.7 Les critères de choix d'un automate programmable	27
Conclusion.....	29

Chapitre II : Logiciel STEP 7 / WinCC

Introduction.....	31
PARTIE A : STEP7, logiciel de programmation des API, Siemens.....	31
II.A.1. Présentation du logiciel STEP7.....	31
II.A.2. Parties essentielle du STEP7.....	31
II.A.2.1. Gestionnaire de projets SIMATIC	31
II.A.2.2. Editeur de mnémoniques.....	32
II.A.2.3. Langages de programmation.....	32
II.A.2.4. Configuration matérielle.....	33
II.A.2.5. Diagnostic du matériel	33
II.A.2.6. Paramétrage de l'interface PG-PC	33
II.A.3. Conception d'une structure programme complète	34
a- Création du projet SIMATIC Step7	34
b- Configuration du matériel.....	35
c- Définition des mnémoniques.....	37
d- Edition des programmes	39
e- Simulation de modules Avec le logiciel PLCSIM	41
f- Chargement du programme dans la CPU	41
PARTIE B : WinCC, progiciel de conception des interfaces HMI.....	42
II.B.1. Présentation de WinCC.....	42
II.B.1.1. Description générale.....	42
II.B.1.2. Fonctionnement du WinCC.....	42
II.B.1.3. Eléments de l'interface utilisateur de WinCC flexible.....	44
II.B.2 Etapes de conception d'un projet sous WinCC.....	48
a. Création d'un projet.....	48
b. Création d'une vue de process	49
c. Définition des variables.....	50
d. Simulation de la vue de process.....	50
II.B.3 Configuration des alarmes.....	51
II.B.4 Création de fonctions et d'actions (Global Script).....	53
Conclusion.....	54

Chapitre III : Description et identification des différents systèmes de CE117

Introduction.....	56
Partie A : Description de la station CE117	56
III.A.1. Description de la station TecQuipmentProcess Trainer CE117.....	56
III.A.2. Les caractéristiques techniques des différents éléments de la station CE117.....	60
III.A.2.1 Les actionneurs	60
III.A.2.2 Les capteurs.....	60
III.A.3. Le logiciel TecQuipmentProcess Trainer CE2000.....	62
III.A.4. Les contraintes de différents systèmes de la station CE117.....	64
Partie B : Identification	65
III.B.1. le but d'identification.....	65
III.B.2. Les méthodes d'identification.....	65
a- Méthodologie.....	65
b-Système naturellement stable.....	66
c- Système naturellement instable.....	67
Partie C : Identification des différents systèmes (régulateur-réglages) du CEE 117 process trainer	68
III.C.1. Le système de régulation de débit.....	68
III.C.1.1. Description du système de régulation de débit.....	68
III.C.1.2. Identification du système pompe-débit.....	69
III.C.1.3. Identification du système vanne – débit.....	70
III.C.2. Le système de régulation de niveau.....	71
III.C.2.1. Description du système de régulation de niveau.....	71
III.C.2.2. Identification du système pompe – niveau.....	71
III.C.2.3. Identification du système vanne – niveau	73
III.C.3. Le système de régulation de pression.....	74
III.C.3.1. Description du système de régulation de pression.....	74
III.C.3.2. Identification du système pompe – pression	75
III.C.3.3. Identification du système vanne – pression	76
III.C.4. Le système de régulation de température.....	78
III.C.4.1. Description du système de régulation de température	78
III.C.4.2. Identification du système pompe-température.....	78
Conclusion.....	78

CHAPITRE IV : Régulation des différents systèmes De la station Process Trainer (CE117)

Introduction.....	80
PARTIE A : Description les commandes utilisées pour la synthèse de commande.....	80
IV.A.1.Commande par retour d'état plus l'action intégrale.....	80
IV.A.2.Commande par mode de glissement	82
IV.A.3.La commande par logique floue.....	84
IV.A.4. La commande adaptative indirecte.....	85
IV.A.5. Commande robuste par loop-shapping	86
PARTIE B: Synthèse de régulateur de différents systèmes.....	90
IV.B.1.Régulation de débit.....	90
a. Système pompe-débit	90
<i>a.1. Régulation par PI</i>	90
<i>a.2.Régulation par mode de glissement</i>	94
<i>a.3.Régulation par Retour d'état plus Action Intégrale</i>	95
b. Système vanne-débit.....	96
<i>b.1.Régulation par PI</i>	96
<i>b.2.Régulation par mode de glissement</i>	96
<i>b.3.Régulation par Retour d'état plus Action Intégrale</i>	96
IV.B.2.Régulation de niveau.....	97
• Résolution de problème de non linéarité du capteur de niveau	97
a. Système pompe-niveau	99
<i>a.1.Régulation par gain proportionnel</i>	99
<i>a.2.Régulation par mode de glissement</i>	100
• Problème de bruit du capteur	102
<i>a.3. commande discrète</i>	105
<i>a.4. Régulation cascade</i>	106
b. Système vanne-niveau.....	109
<i>b.1.Régulation par gain proportionnel</i>	109
<i>b.2.Régulation par mode de glissement</i>	110
<i>b.3.Régulation cascade</i>	110
IV.B.3.Régulation de pression	112
a. Système pompe-pression	112
<i>a.1.Régulation par PI</i>	112
<i>a.2.Régulation par mode de glissement</i>	113

b. Système vanne-pression	114
<i>b.1.Régulation par gain proportionnel</i>	114
<i>b.2.Régulation par mode de glissement</i>	115
<i>b.3.commande discrète</i>	116
IV.B.4.Problèmes de variations paramétriques et des erreurs de modélisations...	117
a. Commande adaptative indirecte	117
b. Régulateur PI avec adaptation paramétrique par la logique floue.....	120
IV.B.5.Régulation de température.....	122
• Régulation par commande discrète.....	122
PARTIE C : Implémentation et supervision des commandes dans logiciel	
STEP7	124
IV.C.1 Implémentation sur Step7.....	124
IV.C.2 Supervision sur WinCC	130
PARTIE D : Etude comparative	133
-Recommandation pour le choix de la meilleure commande	133
Conclusion.....	134
Conclusion générale	135
Perspective	136
Bibliographie	137
Annexe A	139
Annexe B	144

LISTE DES TABLEAUX

Tableau II.1 : Accès aux différentes zones mémoires.....	39
Tableau II.2 : Description des éléments de Menu Sous WinCC flexible.....	45
Tableau III.1 : schéma du module de contrôle de la station CE117.	60
Tableau III.2 : Tableau qui résume la caractéristique du capteur de niveau.....	62
Tableau IV.1 : Les OB d’alarme cyclique.....	130

LISTE DE FIGURES

Figure I.1 : Automate programmable Schneider.....	17
Figure I.2 : Automate programmable compact de Allen-Bradley et modulaire de Modicon....	18
Figure I.3 : Automate programmable modulaire SIEMENS.....	19
Figure I.4 : Le format graphique d'un programme GRAFCET.....	23
Figure I.5 : Le format graphique d'un programme en Ladder.....	24
Figure I.6 : Le format graphique d'un programme en Ladder.....	24
Figure I.7 : Le format graphique d'un programme en FBD.....	25
Figure I.8 : Le format graphique d'un programme en Structuré.....	25
Figure I.9 : Le format graphique d'un programme en Structuré.....	26
Figure I.10 : Cycle de traitement d'un programme par un automate.....	27
FigureII.1 : création d'un nouveau projet.....	34
FigureII.2 : choix de la station de travail.....	35
FigureII.3 :Configuration matériels	35
FigureII.4 :Sélection des modules.....	36
FigureII.5 :Edition des Mnémoniques.	37
FigureII.6 : Edition des programmes.....	39
FigureII.7 : logiciel de simulation PLC-SIM.....	41
Figure II.8 : Structure interne du WinCC.....	43
Figure II.9 : Eléments de l'interface WinCC.....	44
Figure II.10 : La forme graphique d'une Vue.	45
Figure II.11 : fenêtre de projet.	46
Figure II.12 : Fenêtre des propriétés.....	46
Figure II.13 : Elément de la bibliothèque.....	47
Figure II.14 : Fenêtre des erreurs et avertissements.....	47
Figure II.15 : Fenêtre des objets.....	48
Figure II.16 : Fenêtre de sélection du pupitre.....	48
Figure II.17 : Une vue principale.....	49
Figure II.18 : Fenêtre des variables.....	50
Figure II.19 : Fenêtre d'alarme	52
Figure II.20 : Fenêtre classe d'alarme.....	52
Figure II.21 : Global Script.	53

Figure III.1 : la station CE117 Process Trainer.....	57
Figure III.2 : Le module de contrôle de la station CE117.....	58
Figure III.3 : schéma du module de contrôle de la station CE117.....	59
Figure III.4 : La caractéristique statique du capteur de niveau.....	61
Figure III.5 : L'interface de logiciel CE2000.....	62
Figure III.6 : La barre d'outils de logiciel CE2000.....	63
Figure III.7 : la fenêtre de graphe de logiciel CE2000.....	63
Figure III.8 : fenêtre d'options des graphes enregistrés de logiciel CE2000.....	64
Figure III.9 : Les trois grandes familles de courbes.....	65
Figure III.10 : Schéma qui montre la méthode d'identification d'un premier ordre.....	66
Figure III.11 : Schéma qui montre la méthode d'identification d'un intégrateur pur.....	67
Figure III.12 : Schéma qui présente le circuit fluide procédé.....	68
Figure III.13: Réponse réelle et identifiée du système pompe-débit en boucle ouverte.....	69
Figure III.14 : Réponse réelle et identifiée du système vanne-débit en boucle ouverte.....	70
Figure III.15 : Schéma qui présente le circuit fluide procédé.....	71
Figure III.16 : Réponse réelle et identifiée du système pompe-niveau en boucle ouverte.....	72
Figure III.17 : Réponse réelle et identifiée du système vanne-niveau en boucle ouverte.....	74
Figure III.18 : Schéma qui présente le circuit fluide procédé.....	75
Figure III.19 : Réponse réelle et identifiée du système pompe-pression en boucle ouverte.....	76
Figure III.20 : Réponse réelle du système vanne-pression en boucle ouverte.....	77
Figure III.21 : Réponse réelle et identifiée du système vanne-pression en boucle ouverte.....	78
FigureIV.1: Diagramme de bode de transfert en boucle ouvert qui montre les contraintes fréquentielles sur $G(s)$ sous formes des gabarits.....	89
Figure IV.2: diagramme de bode qui montre les erreurs de modélisation.....	92
Figure IV.3 : diagramme de bode qui montre Le gabarit de robustesse.....	93
Figure IV.4: la réponse du système pompe-débit commandé par un PI sous WinCC.....	93
Figure IV.5 : La réponse du système pompe-débit commandé par mode glissant sous WinCC.....	94
Figure IV.6: la réponse du système pompe-débit commandé par un REI sous WINCC.....	95
Figure IV.7: les fonctions d'appartenances des ensembles flous.....	97
Figure IV.8: la réponse du système pompe-niveau après la correction du gain du capteur.....	98
Figure IV.9: la réponse du système pompe-niveau commandé par un gain sous WINCC.....	100
Figure IV.10: la réponse du système pompe-niveau commandé par mode glissant.....	101

Figure IV.11: la réponse du système pompe-niveau commandé par mode glissant sous WinCC avant le filtrage.....	102
Figure IV.12: la réponse du système pompe-niveau commandé par mode glissant sous WinCC après le filtrage.....	104
Figure IV.13: la réponse du système pompe-niveau commandé par la commande discrète sous WINCC.....	105
Figure IV.14 : schéma représentatif de circuit fluide procédé.....	106
Figure IV.15: la réponse du système pompe-niveau commandé en cascade sous WINCC.....	109
Figure IV.16: la réponse du système pompe-pression commandé par un PI sous WinCC.....	113
Figure IV.17: la réponse du système vanne-pression commandé par un gain sous WINCC.....	115
Figure IV.18: la réponse du système vanne-pression commandé par mode glissant	116
Figure IV.19: l’organigramme de la commande discrète du système vanne-pression.....	116
Figure IV.20: la réponse du système vanne-pression commandé par la commande discrète....	117
Figure IV.21 : La réponse du système pompe-débit sous WinCC.....	120
Figure IV.22: les fonctions de répartition des ensembles flous.....	121
Figure IV.23 : La réponse du système pompe-débit sous WinCC.....	122
Figure IV.24: Le bloc de la fonction SCALE.....	125
Figure IV.25: Le bloc de la fonction UNSCALE.....	126
Figure IV.26: La création d’un fichier source.....	126
Figure IV.27: Le choix du modèle de bloc.....	127
Figure IV.28: Programme sous le langage structuré dans un fichier source	127
Figure IV.29: La fenêtre des erreurs de compilation.....	128
Figure IV.30: Bloc de régulation appelé dans un OB1.....	128
Figure IV.31: création du bloc de donné.....	129
Figure IV.32 : La fenêtre de choix des paramètres de réglage.....	130
Figure IV.33 : La vue principale de l’interface graphique IHM.....	131
Figure IV. 34: La vue secondaire de l’interface graphique IHM.....	132

Introduction générale

En vue de l'avancement des systèmes d'automatisations, et de la place qu'occupent dans le monde industrielle. Le département d'automatique à proposer de commander la station CE117 par l'automate programmable S7 314 IFM dans le but de former des élèves ingénieurs automaticiens à maîtriser des systèmes d'automatisation, et à appliquer les aspects théoriques sur des systèmes réels qui ressemble à ceux qui sont disponible dans l'industrie.

Ce travail comporte des parties expérimentales, qui permettent non seulement de voir concrètement l'aboutissement des résultats et la finalité de l'étude, mais aussi de faire ressortir les problèmes cruciaux de la mise en œuvre. Ce travail permet aussi aux étudiants de se familiarisé avec les systèmes industrielles, notamment les systèmes hydraulique et les API, et d'appliquer les connaissances acquises dans l'informatique industrielle, identification, et commande grâce aux différents systèmes de régulation qui se trouve sur la station CE117, tel que le système de régulation de niveau, de débit, de pression, et de température.

Nous exposons dans le présent rapport quatre grands chapitres décrivant les volets principaux de notre projet :

Le premier chapitre sera consacré à la description des automates programmable : domaine d'application, architecture, langages de programmation...etc. et on finira par la description des réseaux de terrain MPI.

Dans le deuxième chapitre, nous décrirons le logiciel Step7 et WinCC flexible de la firme Siemens qui nous permettent de programmer et de contrôler facilement le processus à l'aide d'une l'interface graphique.

Le troisième chapitre sera partager en deux parties la première sera une description de la station CE117 et de logiciel CE2000 utilisé pour l'identification, la deuxième parties sera consacrer à l'identification des différents systèmes de la station (fonctions de transferts et modèles d'états).

Le quatrième chapitre sera partager en quatre parties, la première partie est une partie consacrer à la description des commandes qu'on a l'intention d'utiliser, dans la deuxième partie on synthétisera des différentes commandes pour chaque système de la station, et dans la troisième partie on va présenter brièvement la méthode d'implémentation des commandes par le Step7 et la supervision par le WinCC, et on finira par une étude comparative entre les différentes commandes qui nous permettra de choisir les commandes les plus adéquates pour chaque système.

CHAPITRE I

Les automates programmables industriels

Introduction :

Les automates programmables depuis leur apparition ont bouleversé le monde industriel, maintenant Tous les secteurs de l'industrie, de la production d'électricité à la peinture utilisent les automates programmables industriels pour améliorer la production. Dans ce chapitre, on va présenter les différents aspects de ces outils puissants et polyvalents. On va voir aussi le logiciel de diagnostic de processus S7-PDIAG.

PARTIE A : Description des automates programmables

I.A.1. Historique : [2]

Les automatismes séquentiels ont été réalisés, depuis longtemps, à base de relais électromagnétiques. L'inconvénient c'est qu'il s'agit d'un système câblé ce qui impose la refonte complète du câblage et ceci pour la moindre modification dans l'ordonnancement des séquences. En 1966, l'apparition des relais statiques a permis de réaliser des divers modules supplémentaires tel que le comptage, la temporisation, le pas à pas ... Cependant cette technologie avait le même problème : technologie câblée.

En 1968 et à la demande de l'industrie automobile nord-américaine, sont apparus les premiers dispositifs de commande logique aisément modifiable : Les PLC (Programmable Logic

Controller) par *Allen Bradley, Modicom et Digital Equipement*. Le premier dispositif français était le PB6 de Merlin Gerin en 1973.

I.A.2. Définition :

L'automate programmable, souvent appelé automate programmable industriel (API, en anglais PLC pour Programmable Logic Controller) est défini suivant comme appareil électronique qui comporte une mémoire programmable par un utilisateur automaticien (et non informaticien) à l'aide d'un langage adapté, pour le stockage interne des instructions composant les fonctions d'automatisme comme par exemple: logique séquentiel et combinatoire ; temporisation, comptage, décomptage, comparaison ; calcul arithmétique ; réglage, asservissement, régulation, etc. pour commander, mesurer et contrôler au moyen de modules d'entrées et de sorties (logiques, numériques ou analogiques) différentes sortes de machines ou de processus, en environnement industriel . [2]

C'est donc une machine électronique qui se place entre deux grands courants : la logique câblée et le calculateur universel. Conçu pour fonctionner dans des ambiances industrielles qui peuvent être sévères (poussières, température, humidité, vibrations, parasites électromagnétiques, ...), gérer un grand nombre de signaux d'E/S en temps réel; il reçoit des données par ses entrées, celles-ci sont ensuite traitées par un programme défini, le résultat obtenu étant délivré par ses sorties. Ce cycle de traitement est toujours le même, quel que soit le programme, néanmoins le temps d'un cycle d'API varie selon la taille du programme et la puissance de l'automate. Il dispose de langages adaptés aux fonctions d'automatismes et qui ne réclament pas de connaissances particulières en informatique (programmation simple); flexibles et montage rapide (structure modulaire).



Figure I.1 : Automate programmable Schneider

I.A.3. Domaine d'application :

On trouve les API dans tous les secteurs industriels pour la commande des machines (Convoyage, emballage...) ou des chaînes de production (automobile, agroalimentaire...) il peut également assurer des fonctions de régulation de processus (métallurgie, chimie...). Il est de plus en plus utilisé dans le domaine du bâtiment (tertiaire et industriel) pour le contrôle du chauffage, de l'éclairage, de la sécurité ou des alarmes

I.A.4. L'architecture :

Les automates peuvent être de type compact ou modulaire.

- De type compact, on distinguera les *modules de programmation* (LOGO de Siemens, ZELIO de Schneider, MILLENIUM de Crouzet ...) des *micros automates*. Il intègre le processeur, l'alimentation, les entrées et les sorties. Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage rapide, E/S analogiques ...) et recevoir des extensions en nombre limité. Ces automates, de fonctionnement simple, sont généralement destinés à la commande de petits automatismes. [1]



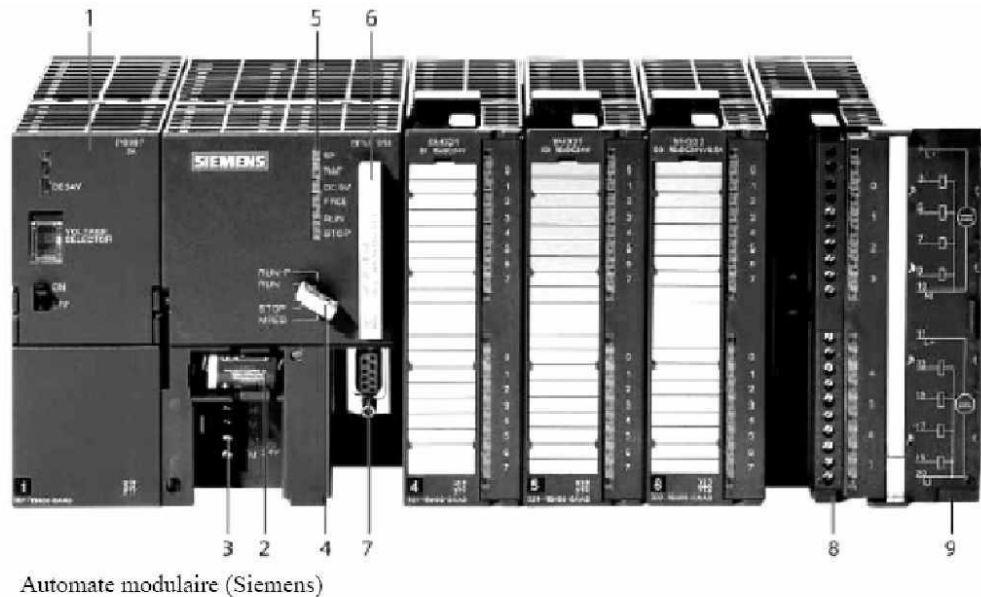
Automate compact (Allen-bradley)



Automate modulaire (Modicon)

Figure I.2 : Automate programmable compact de Allen-Bradley et modulaire de Modicon

- De type modulaire, le processeur, l'alimentation et les interfaces d'entrées / sorties résident dans des unités séparées (modules) et sont fixées sur un ou plusieurs racks contenant le "fond de panier" (bus plus connecteurs). Ces automates sont intégrés dans les automatismes complexes où puissance, capacité de traitement et flexibilité sont nécessaires. [1]



Automate modulaire (Siemens)

- | | | | |
|---|---|---|----------------------------|
| 1 | Module d'alimentation | 6 | Carte mémoire |
| 2 | Pile de sauvegarde | 7 | Interface multipoint (MPI) |
| 3 | Connexion au 24V cc | 8 | Connecteur frontal |
| 4 | Commutateur de mode (à clé) | 9 | Volet en face avant |
| 5 | LED de signalisation d'état et de défauts | | |

Figure I.3 : Automate programmable modulaire SIEMENS

En général, les automates sont conçus pour être modulaires, notamment pour pouvoir augmenter le nombre d'E/S. D'où l'utilisation d'une structure d'un rack dans lequel s'encastrent les différentes cartes (modules) : module d'alimentation, l'unité centrale, coupleurs, module d'entrées/sorties, module ou processeur de communication module de fonction et des auxiliaires.

a- Module d'alimentation "PS":

Il est composé de blocs qui permettent de fournir à l'automate l'énergie nécessaire à son fonctionnement, il convertit la tension du réseau (AC 220 V) en tension de service (DC 24V, 12V ou 5V) et assure l'alimentation de l'automate ainsi que les autres modules. Un voyant est positionné en générale sur la façade pour indiquer la mise sous tension de l'automate.

b- L'unité centrale "CPU":

L'unité centrale (CPU) est l'élément le plus important dans l'automate programmable, elle peut être considérée comme le cerveau du système. À base d'un processeur et de mémoires, elle réalise toutes les fonctions logiques, arithmétiques et de traitement numérique (transfert, comptage, temporisation ...).

b.1- Processeur :

Appelé unité de traitement, il assure le contrôle de l'ensemble de la machine et effectue les traitements demandés par les instructions du programme (exécution d'un ensemble de programmes utilisateur et système). Il réalise les fonctions logiques, temporisation, comptage, calcul. Il comporte un certain nombre de registres (compteur ordinal, registre d'instructions, registre d'adresse, registres de données, accumulateurs. Il est connecté aux autres éléments (mémoires, interfaces d'E/S, ...) par l'intermédiaire des bus. [2]

b.2- Mémoire :

La mémoire d'un automate est l'élément fonctionnel qui peut recevoir, conserver et restituer l'information. Elles permettent de stocker le système d'exploitation (ROM ou PROM), le programme (EEPROM) et les données système lors du fonctionnement (RAM). Cette dernière est généralement secourue par pile ou batterie. On peut, en règle générale, augmenter la capacité mémoire par adjonction de barrettes mémoires type. Pour un automate, il faut connaître la capacité mémoire minimale utile et la capacité maximale que l'on peut obtenir par diverses extensions. [1]

c- Coupleurs "IM":

Ce sont des cartes électroniques qui assurent la communication entre les périphériques (modules d'E/S ou autres) et l'unité centrale. Les API peuvent être connectées entre eux par des coupleurs spécialisés. Les informations sont alors transmises de coupleur à coupleur par l'intermédiaire d'un bus. Ils sont utilisés aussi pour la connexion des racks d'extension ex : ET200. [2]

d- Les modules d'entrées/sorties "SM":

Le module E/S assure le rôle d'interface pour la partie commande, qui distingue la partie opérative (les sorties), où les actionneurs agissent physiquement sur le processus, et la partie d'acquisitions (les entrées) récupérant les informations sur l'état de ce processus et coordonnant en conséquence les actions pour atteindre les objectifs prescrits (matérialisés par des consignes). En plus d'assurer la communication entre la CPU et les organes externes, le module d'E/S doit garantir une protection contre les parasites électriques, c'est pourquoi la plus part des modules E/S font appel au découplage optoélectroniques.

d.1 Les informations traitées par l'automate:

Les informations traitées par l'automate peuvent être de type :

1. Tout ou rien (T.O.R.) : l'information ne peut prendre que deux états (vrai/faux, 0 ou 1). C'est le type d'information délivrée par un détecteur, un bouton poussoir ...
2. Analogique : l'information est continue et peut prendre une valeur comprise dans une plage bien déterminée. C'est le type d'information délivrée par un capteur (pression, température ...)
3. Numérique : l'information est contenue dans des mots codés sous forme binaire ou bien hexadécimale. C'est le type d'information délivrée par un ordinateur ou un module intelligent.[1]

d.2 Les caractéristiques des entrées et des sorties d'un API :

Les caractéristiques des entrées sont :

1. Le nombre et la nature (TOR, numérique, analogique, etc.).
2. Spécifications électriques de raccordement (tension, courant, alimentation).
3. Filtrage, c'est-à-dire : capacité à ne pas laisser passer les parasites ou signaux d'une durée inférieure à une valeur définie. Les valeurs standard vont jusqu'à quelques dizaines de millisecondes.

Les caractéristiques des sorties sont :

1. nombre et nature.
2. technologie : à contact mécanique (relais) ou statique (composant électronique), et les temps de commutation associés (de la milliseconde pour les contacts à quelques dizaines de microsecondes pour les transistors).
3. spécifications électriques de raccordement (tension, courant, puissance, etc.).[2]

e- Le module de fonction "FM" (Les cartes spécialises) :

Les automates modulaires permettent de réaliser de nombreuses fonctions grâce à des modules intelligents appelé module de fonction que l'on dispose sur un ou plusieurs racks. Ces modules ont l'avantage de ne pas surcharger le travail de la CPU car ils disposent bien souvent de leur propre processeur. Parmi ces modules on peut citer : les cartes de commande d'axe, les concentrateurs de communication, les cartes E/S déportées, les cartes de comptage rapide, les cartes de pesage, les cartes de régulations PID, etc.

f- Processeur de communication "CP":

Le module de communication c'est un module utilisé pour ajouter un autre mode de communication (PROFIBUS, Industriel Ethernet, . . .) avec l'API.

g- Les auxiliaires :

Il s'agit principalement de :

1. Un support mécanique (un rack) : l'automate se présente alors sous forme d'un ensemble de cartes, d'une armoire, d'une grille et des fixations correspondantes.
2. Un ventilateur : il est indispensable dans les châssis comportant de nombreux modules ou dans le cas où la température ambiante est susceptible de devenir assez élevée (plus de 40C).
3. Un indicateurs d'état : il indique la présence de tension, l'exécution du programme (mode RUN), la charge de la batterie, le bon fonctionnement des coupleurs.

I.A.5. Langages de programmation :

La norme IEC 1131-3 (Commission Électrotechnique Internationale) définit cinq langages qui peuvent être utilisés pour la programmation des automates programmables industriels. Ces langages peuvent être divisés en deux catégories :

a- Langages graphiques :

- SFC « Sequential Funiculite Chart » ou GRAFCET.
- LD « Ladder Diagram » ou schéma à relais.
- FBD « Function Block Diagram » ou schéma par bloc.

b- Langages textuels :

- ST « structured text » ou texte structuré.
- IL « Instruction List » ou liste d'instructions.

a. Les langages graphiques :

a.1. Le Grafcet « SFC »:

Le GRAFCET ou Graphe Fonctionnel de Commande Etape Transition est une méthode de représentation graphique permettant de décrire le cahier de charge d'un automatisme. Il est adapté aux systèmes à évolution séquentielle ; il est défini par un ensemble d'éléments graphiques de base traduisant le comportement de la partie commande

vis-à-vis de ses entrées et ses sorties.

Un programme GRAFCET décrit un procédé comme une réceptivité .Celle-ci est une condition logique qui doit être vraie pour franchir la transition et passer à l'étape suivante. Des actions sont associées aux étapes du programme.

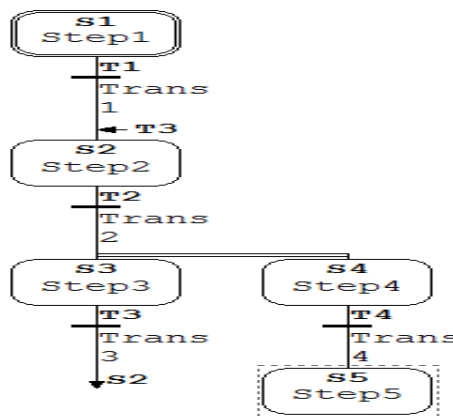


Figure I.4 : Le format graphique d'un programme GRAFCET

Une étape représentée par un carré qui a un numéro identificateur et les actions associées sont indiquées dans un rectangle relié à la partie droite du carré (l'étape initiale est représentée par un carré double). Une liaison orientée représentée par une ligne, parcourue par défaut de haut en bas ou de gauche à droite. Une transition entre deux étapes et à laquelle est associée une réceptivité inscrite à sa droite, est représentée par une barre perpendiculaire aux liaisons orientées qui relient ces étapes.

a.2. Ladder Diagram « CONT »:

Le LD est une représentation graphique qui traduit directement des équations booléennes en un circuit électrique et ce en combinant des contacts et des relais à l'aide de connexions horizontales et verticales ; les contacts représentent les entrées (contact normalement ouverts, contacts

normalement fermés, ...) et les relais les sorties (relais directs, relais inversés,...). Les diagrammes LD sont limités sur la gauche par une barre d'alimentation et par la masse sur la droite.

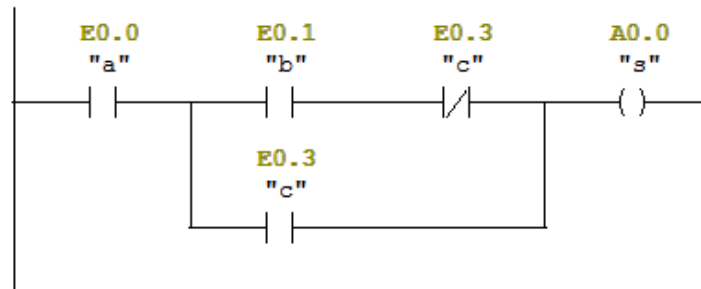


Figure I.5 : Le format graphique d'un programme en Ladder

Le langage LD propose d'autres types de fonction tel que les fonctions de comptages et de temporisations, les fonctions arithmétiques et logiques, les fonctions de comparaison et de transfert.

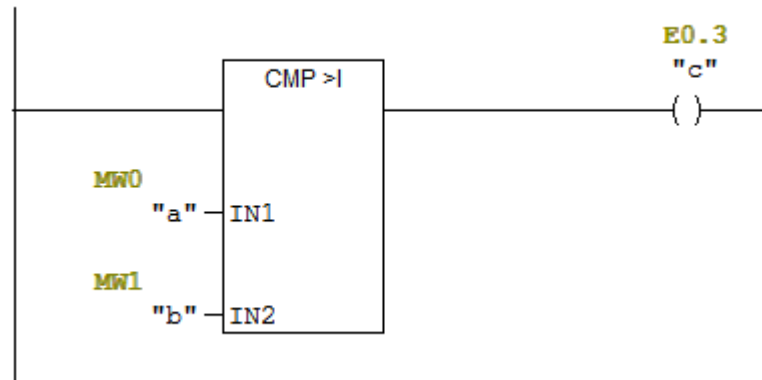


Figure I.6 : Le format graphique d'un programme en Ladder

a.3. Bloc fonction diagramme « FBD » :

C'est un langage graphique qui permet la construction d'équations complexes à partir des opérateurs standards, ou de blocs fonctionnels ; il se compose de réseaux de fonctions préprogrammées ou non, représentées par des rectangles connectés entre eux par des lignes.

La programmation avec le FBD est très souple et facile à apprendre, la plupart des fonctions nécessaires (les fonctions arithmétique et logique, les fonctions de temporisation, des blocs fonctionnels PID...) sont déjà disponibles dans la bibliothèque. Il suffit juste de les connecter et de

bien paramétrer les entrées et les sorties, c'est-à-dire respecter le type des variables lors de la connexion.

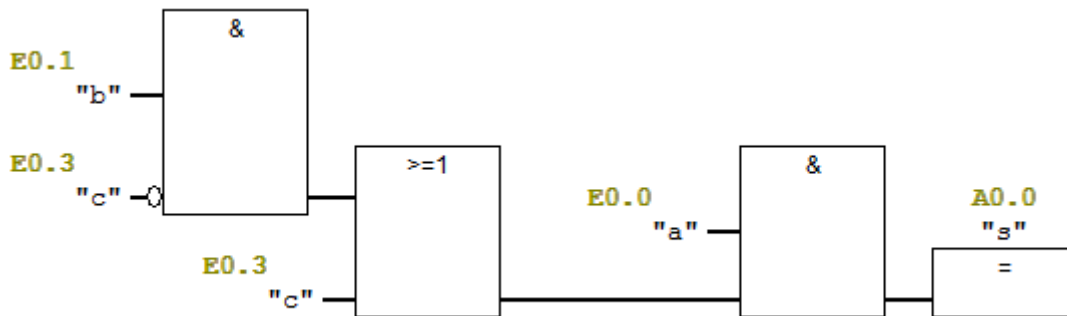


Figure I.7 : Le format graphique d'un programme en FBD

b. Les langages textuels :

b.1. Texte Structuré « SCL » :

Le langage ST (Structured Text) est un langage de programmation textuel de haut niveau dédié aux applications d'automatisation ; il est utilisé principalement pour décrire les procédures complexes et difficilement modélisables avec les langages graphiques. Il peut aussi être utilisé en tant que sous-programme avec d'autres langages de programmation. Il utilise les mêmes énoncés que les langages de programmation de haut niveau (Pascal, C, C++...) comme: les assignations, les appels de fonction, les énoncés de contrôle (IF, THEN, ELSE, CASE) ou d'itération (FOR, WHILE, REPEAT), en plus des opérations arithmétiques et logiques.

```

FUNCTION_BLOCK FB1

VAR_INPUT
  a:REAL;
  b:REAL;
END_VAR
VAR_OUTPUT
  s:REAL;
END_VAR
BEGIN
  s:=a+b;
END_FUNCTION_BLOCK

```

Figure I.8 : Le format graphique d'un programme en Structuré

b.2. Liste d'Instructions « IL »:

Le langage IL est un langage textuel de bas niveau (proche du langage machine), qui utilise un jeu d'instructions simples. Il trouve sa puissance dans les applications de petites tailles, et dans la création de sous-programme ou procédure, car il permet un contrôle totale et une optimisation parfaite du code ; par contre pour les grandes applications il est très difficile de programmer avec le IL ; les programmes dans ce langage peuvent être traduit ou déduit des autres langages.

Le IL a la même structure que l'assembleur ; il utilise un ou plusieurs registres de travail. Les valeurs intermédiaires nécessaires pour l'exécution d'une instruction donnée seront mémorisées dans ces registres le temps de leur utilisation et il possède un jeu d'instructions assez riche pour décrire toutes les opérations arithmétiques et logiques, les opérations de comptage et temporisation, la comparaison et le transfert...

```

U      "a"          E0.0
U(
U      "b"          E0.1
UN    "c"          E0.3
O     "c"          E0.3
)
=     "3"          A0.0

```

Figure I.9 : Le format graphique d'un programme en Structuré

I.A.6. Traitement du programme par automate : [1]

Tous les automates fonctionnent selon le même mode opératoire :

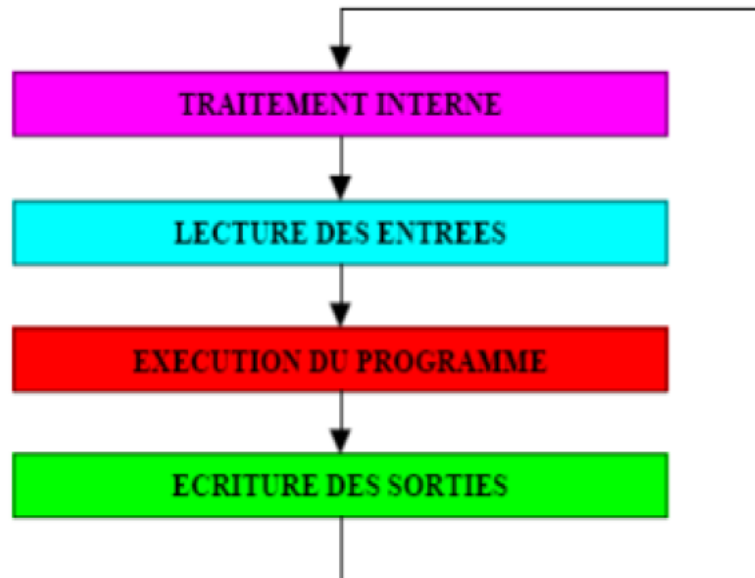


Figure I.10 : Cycle de traitement d'un programme par un automate

a. Traitement interne :

L'automate effectue des opérations de contrôle et met à jour certains paramètres systèmes (détection des passages en RUN / STOP, mises à jour des valeurs de l'horodateur, ...).

b. Lecture des entrées :

L'automate lit les entrées (de façon synchrone) et les recopie dans la mémoire image des entrées.

c. Exécution du programme :

L'automate exécute le programme instruction par instruction et écrit les sorties dans la mémoire image des sorties.

d. Ecriture des sorties :

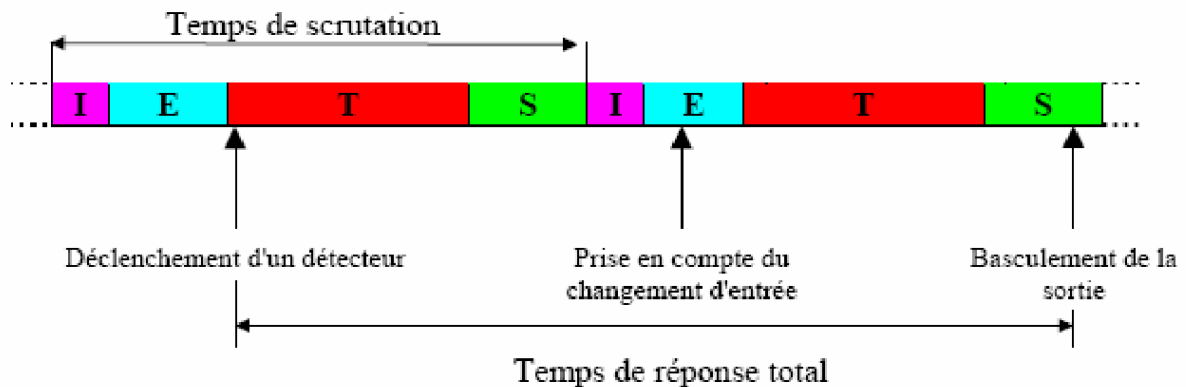
L'automate bascule les différentes sorties (de façon synchrone) aux positions définies dans la mémoire image des sorties. Ces quatre opérations sont effectuées continuellement par l'automate (fonctionnement cyclique). On appelle scrutation l'ensemble des quatre opérations réalisées par l'automate et le temps de scrutation est le temps mis par l'automate pour traiter la même partie de programme. Ce temps est de l'ordre de la dizaine de millisecondes pour les applications standards.

[1]

Le temps de réponse total (TRT) : [1]

Est le temps qui s'écoule entre le changement d'état d'une entrée et le changement d'état de la sortie correspondante :

Le temps de réponse total est au plus égal à deux fois le temps de scrutation (sans traitement particulier).



Le temps de scrutation est directement lié au programme implanté. Ce temps peut être fixé à une valeur précise (fonctionnement périodique), le système indiquera alors tout dépassement de période. Dans certains cas, on ne peut admettre un temps de réponse aussi long pour certaines entrées : ces entrées pourront alors être traitées par l'automate comme des événements (traitement événementiel) et prises en compte en priorité (exemples : problème de sécurité, coupure d'alimentation ...). Certains automates sont également pourvus d'entrées rapides qui sont prises en compte avant le traitement séquentiel mais le traitement événementiel reste prioritaire. Exemple : Les automates TSX micro(Télémechanique) offrent deux types de structure logicielle :

Une structure mono tâche : Le programme n'est alors lié qu'à une seule tâche : la tâche maîtresse.

Une structure multitâche : A la tâche précédente peut être rajouté deux autres tâches : la tâche rapide et la tâche événementielle. La tâche rapide est alors périodique pour laisser le temps à la tâche maîtresse de s'exécuter (la moins prioritaire). La tâche événementielle est prioritaire sur les autres tâches.

I.A.7 Les critères de choix d'un automate programmable :

Le choix d'un automate programmable est en premier lieu le choix d'une société ou d'un groupe et les contacts commerciaux et expériences vécues sont déjà un point de départ. Les grandes sociétés privilégieront deux fabricants pour faire jouer la concurrence et pouvoir "se retourner" en cas de "perte de vitesse" de l'une d'entre elles. Le personnel de maintenance doit toutefois être

formé sur ces matériels et une trop grande diversité des matériels peut avoir de graves répercussions.

Un automate utilise des langages de programmation de type GRAFCET est également préférable pour assurer les mises au point et dépannages dans les meilleures conditions. La possession d'un logiciel de programmation est aussi source d'économies (achat du logiciel et formation du personnel). Des outils permettant une simulation des programmes sont également souhaitables.

Il faut ensuite quantifier les besoins :

1. Nombre d'entrées / sorties : le nombre de cartes peut avoir une incidence sur le nombre de racks dès que le nombre d'entrées / sorties nécessaires devient élevé.
2. Type de processeur : la taille mémoire, la vitesse de traitement et les fonctions spéciales offertes par le processeur permettront le choix dans la gamme souvent très étendue.
3. Fonctions ou modules spéciaux : certaines cartes (commande d'axe, pesage ...) permettront de "soulager" le processeur et devront offrir les caractéristiques souhaitées (résolution, ...).
4. Fonctions de communication : l'automate doit pouvoir communiquer avec les autres systèmes de commande (API, supervision ...) et offrir des possibilités de communication avec des standards normalisés (PROFIBUS ...). [1]

Conclusion :

La plus part des grands constructeurs d'automates programmables, fournissent des logiciels de configuration et de programmation munis des langages SFC, LD, FBD, ST et IL.

Le choix d'un langage s'appuie sur la complexité de l'application et de la tâche de commande. Il est préférable d'utiliser les langages graphiques (SFC, LD et FBD) pour la réalisation des programmes de commande séquentielle. Le SFC est la réalisation direct d'un GRAFCET de commande, les langages LD et FBD sont plus utiles pour les opérations combinatoires sur bits ou mots.

Les langages textuels sont beaucoup plus performants pour le traitement de variables continues ou analogiques ainsi que pour la commande des systèmes continus. Les programmes en IL sont un peu fastidieux à mettre en œuvre, mais connaissent une optimisation optimale pour le temps de traitement et l'occupation de la mémoire. Le ST est le langage par excellence, très utile pour des utilisateurs ayant des connaissances en langages évolués tel que PASCAL.

CHAPITRE II

**STEP7, logiciel de programmation des API
Siemens**

**WinCC, progiciel de conception des interfaces
HMI**

Introduction :

Munis de deux logiciels très performants (STEP7, WinCC), les automates programmables de la famille S7-300/S7-400 forment des unités de contrôle puissantes et flexibles. En effet, simple à utiliser et doté d'une interface graphique très intuitive.

Dans ce chapitre on va présenter en détaille les deux logiciels cité précédemment, le STEP7 pour la programmation et la simulation, le WinCC pour la supervision des procédés automatisés.

PARTIE A : STEP7, logiciel de programmation des API, Siemens

II.A.1. Présentation du logiciel STEP7 : [1]

STEP 7 est le progiciel de base pour la configuration et la programmation de systèmes d'automatisation SIMATIC. Il fait partie de l'industrie logicielle SIMATIC. Le progiciel de base STEP 7 existe en plusieurs versions :

- STEP 7-Micro/DOS et STEP 7-Micro/Win pour des applications autonomes simples sur SIMATIC S7 - 200.
- STEP 7 pour des applications sur SIMATIC S7-300/400, SIMATIC M7-300/400 et SIMATIC C7 présentant des fonctionnalités supplémentaires :
 - Possibilité d'extension grâce aux applications proposées par l'industrie logicielle SIMATIC (voir aussi Possibilités d'extension du logiciel de base STEP 7)
 - Possibilité de paramétrage de modules fonctionnels et de modules de communication
 - Forçage et fonctionnement multiprocesseur
 - Communication par données globales
 - Transfert de données commandé par événement à l'aide de blocs de communication et de blocs fonctionnels
 - Configuration de liaisons

STEP 7 fait l'objet du présent manuel d'utilisation, STEP 7-Micro étant décrit dans la documentation "STEP 7-Micro/DOS".

II.A.2. Parties essentielle du STEP7:

II.A.2.1. Gestionnaire de projets SIMATIC :

Le gestionnaire de projets SIMATIC gère toutes les données relatives à un projet d'automatisation quel que soit le système cible (S7/M7/C7) sur lequel elles ont été créées. Le gestionnaire des projets SIMATIC démarre automatiquement les applications requises pour le traitement des données sélectionnées.

II.A.2.2. Editeur de mnémoniques :

Il permet de gérer toutes les variables globales. C'est à dire la définition de désignations symboliques et de commentaires pour les signaux du processus (entrées/sorties), mémentos et blocs, l'importation et l'exportation avec d'autres programmes Windows.

II.A.2.3. Langages de programmation :

Les langages de programmation CONT, LIST et LOG pour S7-300/400 font partie intégrante du logiciel de base.

- **Le schéma à contacts (CONT) :**

Est un langage de programmation graphique. La syntaxe des instructions fait penser aux schémas de circuits. CONT permet de suivre facilement le trajet du courant entre les barres d'alimentation en passant par les contacts, les éléments complexes et les bobines.

- **La liste d'instructions (LIST) :**

Est un langage de programmation textuel proche de la machine. Dans un programme LIST, les différentes instructions correspondent, dans une large mesure, aux étapes par lesquelles la CPU traite le programme. Pour faciliter la programmation, LIST a été complété par quelques structures de langage évolué (comme, par exemple, des paramètres de blocs et accès structurés aux données).

- **Le logigramme (LOG) :**

Est un langage de programmation graphique qui utilise les boîtes de l'algèbre de Boole pour représenter les opérations logiques. Les fonctions complexes comme par exemple les fonctions mathématiques, peuvent être représentées directement combinées avec les boîtes logiques. Nous disposons des logiciels de langage optionnels (langage évolué) suivants pour la programmation des automates programmables SIMATIC S7-300/400.

- **GRAPH :**

Est un langage de programmation permettant la description aisée de commandes séquentielles (programmation de graphes séquentiels). Le déroulement du processus y est subdivisé en étapes. Celles-ci contiennent en particulier des actions pour la commande des sorties. Le passage d'une étape à la suivante est soumis à des conditions de transition.

- **HiGraph :**

Est un langage de programmation permettant la description aisée de processus asynchrones non séquentiels sous forme de graphes d'état. A cet effet, l'installation est subdivisée en unités fonctionnelles pouvant prendre différents états. Ces unités fonctionnelles peuvent se synchroniser par l'échange de messages.

- **SCL :**

Est un langage évolué textuel conforme à la norme DIN EN 61131-3. Il comporte des éléments de langage que l'on trouve également sous une forme similaire dans les langages de programmation Pascal et C. SCL convient donc particulièrement aux utilisateurs déjà habitués à se servir d'un langage de programmation évolué. Nous pouvons, par exemple, faire appel à SCL pour programmer des fonctions très complexes ou se répétant souvent.

Nous disposons aussi des Langages graphiques

- **CFC :**

Pour S7 et M7 est un langage de programmation permettant l'interconnexion graphique de fonctions existantes. Ces fonctions couvrent un large éventail allant de combinaisons logiques simples à des régulations et commandes complexes. Un grand nombre de ces fonctions est disponible sous la forme de blocs dans une bibliothèque. La programmation se fait en copiant des blocs sur un diagramme et en reliant les connecteurs de blocs par des lignes.

II.A.2.4. Configuration matérielle :

Il permet de configurer et paramétrer le matériel d'un projet d'automatisation. Il suffit juste de sélectionner le châssis (Rack) dans un catalogue électronique et leurs affecter les modules sélectionnés aux emplacements souhaités dans les racks (CPU, SM, FM. . .). De plus il permet le paramétrage de la CPU (comportement à la mise en route, surveillance du temps de cycle), des modules fonctionnels (FM) et de processeurs de communication (CP).

II.A.2.5. Diagnostic du matériel :

Le diagnostic du matériel fournit un aperçu de l'état du système d'automatisation. Dans une représentation d'ensemble, un symbole permet de préciser pour chaque module, s'il est défaillant ou pas. Un double clic sur le module défaillant permet d'afficher des informations détaillées sur le défaut.

Les informations disponibles dépendent des différents modules :

- affichage d'informations générales sur le module (p.ex. numéro de commande, version, désignation) et son état (p.ex. défaillant),
- affichage d'erreurs sur les modules (p.ex. erreur de voie) de la périphérie centrale et des esclaves DP,
- affichage des messages de la mémoire tampon de diagnostic.

Pour les CPU, des informations supplémentaires s'affichent :

- causes de défaillance dans le déroulement d'un programme utilisateur
- durée de cycle (le plus long, le plus court et dernier),
- possibilités et charge de la communication MPI,
- performances (nombre d'entrées/sorties, de mémentos, de compteurs, de temporisations et de blocs possibles).

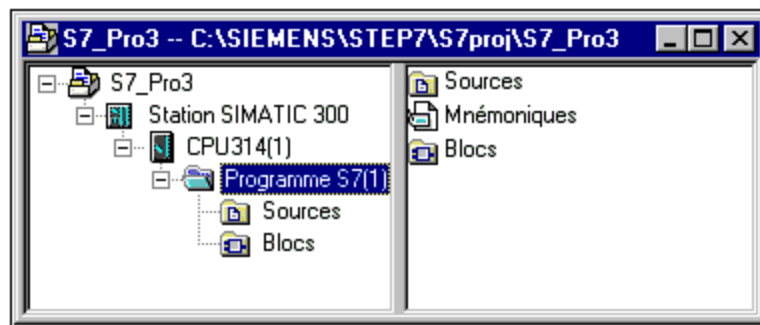
II.A.2.6. Paramétrage de l'interface PG-PC

Cet outil sert à paramétrer l'adresse locale des PG/PC, la vitesse de transmission dans le réseau MPI ou PROFIBUS en vue d'une communication avec l'automate et le transfert du projet.

II.A.3. Conception d'une structure programme complète :

a- Création du projet SIMATIC Step7 :

Un projet comprend deux données essentielles, les programmes et la configuration du matériel, on peut commencer par définir l'une ou l'autre, mais tout d'abord il faut démarrer le programme SIMATIC Manager. Ce programme est l'interface graphique qui permet la manipulation du projet et l'accès aux autres programmes de STEP7.



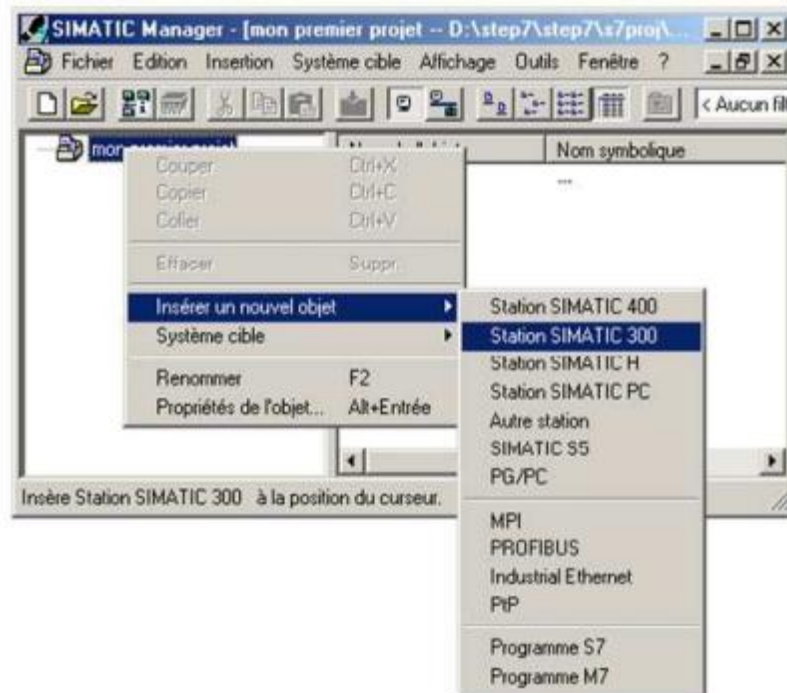
FigureII.1: création d'un nouveau projet

Pour en créer un nouveau, il suffit de cliquer sur le bouton « Nouveau projet », attribuer un nom et valider. Ensuite il faut choisir une station de travail. Une station SIMATIC représente une configuration matérielle S7 comportant un ou plusieurs modules programmables. Il existe différents types:

- SIMATIC 400
- SIMATIC 300
- SIMATIC S5
- SIMATIC H : Automate insensible aux défaillances, il se compose de 2 CPU du même type, en cas de problème elle commute de l'une vers l'autre sans perte de données.
- SIMATIC PC : ou Station PC, représente un PC contenant des composants

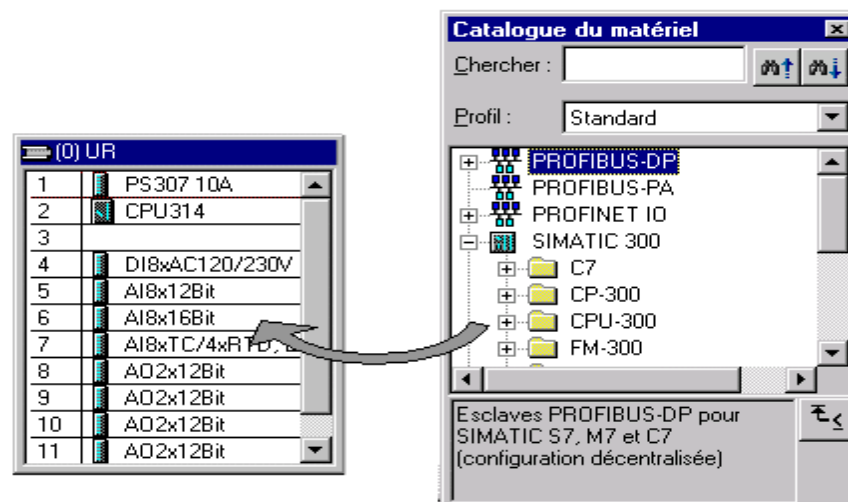
SIMATIC : des applications (WinCC, par ex.), ou une carte CPU enfichée dans le PC.

- PG/PC : Outils de programmation pour contrôleurs SIMATIC, c'est une console de programmation compatible avec le milieu industriel.
- Autres stations : se sont soit des appareils d'autres fabricants ou bien des stations de SIMATIC S7 contenus dans un autre projet.



FigureII.2: choix de la station de travail

Pour commencer, le plus simple est de configurer le matériel, d'éditer les programmes puis les charger dans la CPU. Avec un double clic sur « Matériel », on démarre l'application HW Config.



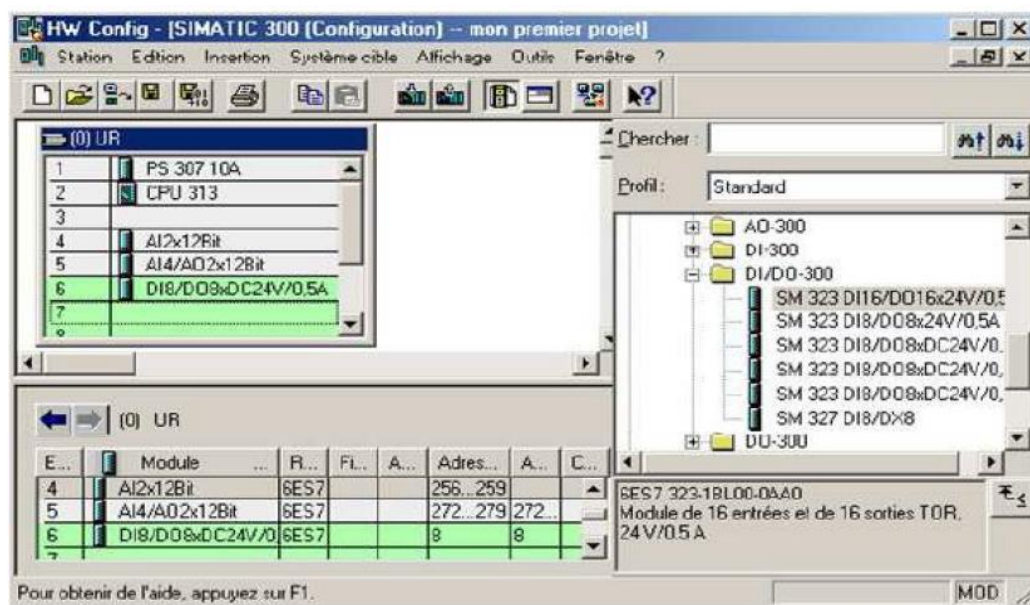
FigureII.3: Configuration matériels

b- Configuration du matériel :

Pour configurer le matériel, il suffit de faire glisser des éléments du catalogue dans l'emplacement approprié, on choisit le Rack, l'alimentation, la CPU et les E/S... Dans le catalogue on trouve les modules qu'on peut affecter à chaque type de station, on distingue:

- C7 : Système intégré compact qui regroupe automate programmable et interface homme machine (pupitre opérateur) pour la réalisation de commandes de machines sous encombrement réduit.

- CP : Communication Processor, module de communication (PROFIBUS, Industriel Ethernet,....).
- FM : (Function Module), il regroupe les modules de fonctions (régulation, comptage...).
- IM : Coupleurs d'extension, il permet l'ajout d'autres modules.
- M7 : Modules d'extension et cartouches interface pour SIMATIC M7.
- PS : Module d'alimentation.
- Rack : Support mécanique.
- Routeur : Relie Industriel Ethernet à PROFIBUS.
- SM : Signal Module, c'est le module d'E/S, il contient le AI module d'entrées analogiques, le AO module de sorties analogiques, le DI module d'entrées TOR et le DO module de sorties TOR.
- CPU : L'Unité Centrale, noté CPU xxx a b.
 - xxx est la famille de la CPU
 - a, b sont les propriétés de la CPU (éléments additionnels, port de communication...). Par exemple :
 - C : compact, la CPU intègre des modules E/S ainsi que des fonctions spécialisées.
 - PtP : Peer to Peer, la CPU intègre un port de communication Point to Point.
 - H : Fault-tolerant, des unités de traitements insensibles aux défaillances
 - DP : Decentralized Periphery, la CPU intègre un port de communication PROFIBUS.



FigureII.4: Sélection des modules

Si l'insertion de l'élément choisi est possible dans le Rack, la case appropriée devient verte. Une fois le matériel choisi on sauvegarde, on compile et on charge dans la CPU.

c- Définition des mnémoniques :

De retour dans le SIMATIC Manager, on trouve de nouveaux éléments. On commence par créer les mnémoniques dans la section programmes.

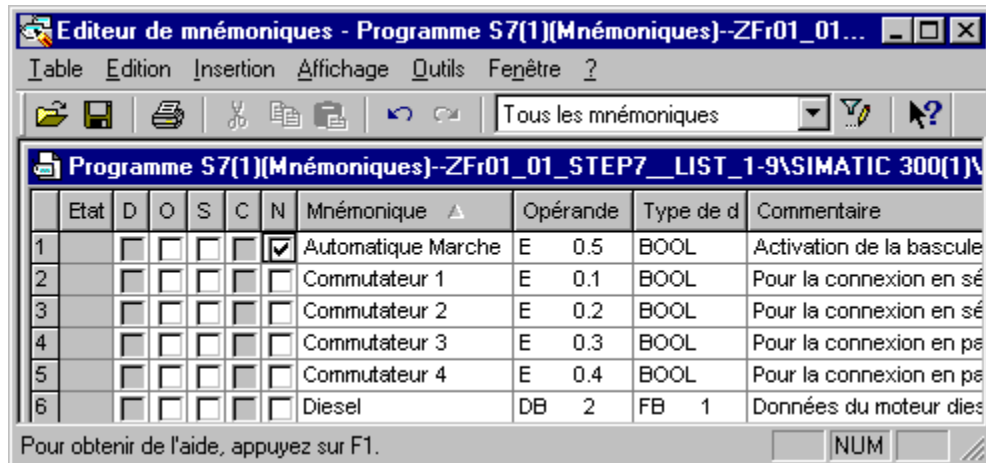


Figure II.5: Edition des Mnémoniques.

En affectant des noms symboliques aux adresses absolues, les programmes deviennent plus lisibles, faciles à corriger et à mettre à jour.

Il y a quatre différents types d'opérande : le bit, l'octet, le mot et le double mot. Ces types définissent l'accès à une zone mémoire. Pour chaque opérande un certain type de données est permis :

- Pour le bit : BOOL : variable booléenne (True ou False, 1 ou 0).
- Pour l'octet : deux types de données sont possibles :
 1. BYTE: nombre hexadécimal de B#16#0 à B#16#FF.
 2. CHAR : Caractère ASCII, 'A', 'B'...
- Pour le mot : quatre types de données sont possible :
 1. WORD : nombre hexadécimal de W#16#0 à W#16#FFFF.
 2. INT : nombre entier de -32768 à 32767.
 3. S5TIME : Durée S7 en pas de 10 ms (valeur par défaut), de S5T#0H 0M 0S 10MS à S5T#2H 46M 30S 0MS.
 4. DATE : Date en incréments de 1 jour, de D#1990-1-1 à D#2168-12-31.

• Pour le double mot : cinq types de données :

1. **DWORD**: nombre hexadécimal de DW#16#0000_0000 à DW#16#FFFF_FFFF.
2. **DINT** : nombre entier de L#-2147483648 à L#2147483647.
3. **REAL** : nombre à virgule flottante, Limite supérieure : 3.402823e+38 Limite inférieure : 1.175 495e-38.
4. **TIME** : Durée en incréments de 1 ms, de -T#24D_20H_31M_23S_648MS à T#24D_20H_31M_23S_647MS.
5. **TIME_OF_DAY** : Heure en pas de 1ms, de TOD#0:0:0.0 à TOD#23:59:59.999.

Tout type de variable peut être associé à une mnémonique dont l'accès est défini comme suit :

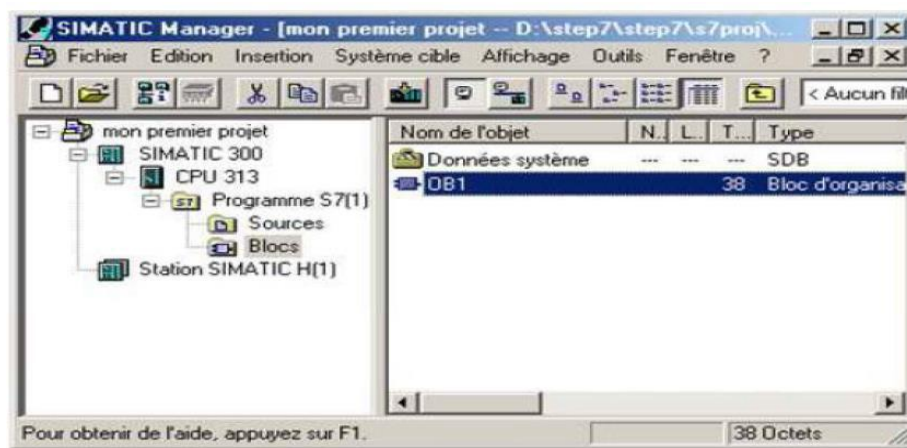
Nom de la zone	Description	Accès à la zone par :
Mémoire Image des Entrées (MIE)	Au début du cycle le système d'exploitation lit les entrées provenant du processus et enregistre ces valeurs dans la MIE. Le programme utilise ces valeurs pendant son traitement normal.	Bit : E Octet : EB Mot : EW Double mot : ED
Mémoire Image des Sorties (MIS)	Pendant le cycle le programme calcule les valeurs de sortie et les dépose dans la MIS. A la fin du cycle le système d'exploitation lit les valeurs de sorties figurées dans la MIS et les transmet aux sorties du processus.	Bit : A Octet : AB Mot : AW Double mot : AD
Mémentos	Ce sont des zones mémoires qui permettent de sauvegarder les résultats intermédiaires calculés dans le programme.	Bit : M Octet : MB Mot : MW Double mot : MD
Périphérie d'entrée et de sortie	Cette zone permet d'accéder directement aux modules d'entrées et de sorties.	Octet : PEB, PAB Mot : PEW, PAW Double mot : PED ou PAD
Temporisations	Cette zone sert d'espace mémoire pour les cellules de temporisation, l'horloge accède à ces cellules afin de les mettre à jour en décrémentant la valeur de temps.	T
Compteurs	Cette zone mémoire, sert d'espace mémoire pour les opérations de comptage.	Z

<p>Données locales</p>	<p>Cette zone contient les données temporaires, elle est utilisée dans les blocs de code (OB, FB ou FC). Ces données sont rangées dans la pile des données locales, elles seront perdues une fois le bloc de code achevé.</p>	<p>Bit : L Octet : LE Mot : LW Double mot : LD</p>
-------------------------------	---	---

Tableau II.1 : Accès aux différentes zones mémoires

d- Edition des programmes :

Dans la section ‘bloc’ du SIMATIC Manager, on trouve par défaut le bloc d’organisation 1 ‘OB1’ qui représente le programme cyclique. On peut rajouter d’autres blocs à tout moment par un clic droit dans la section Bloc de SIMATIC Manager.



FigureII.6: Edition des programmes

Deux programmes différents s'exécutent dans la CPU: le système d'exploitation et le programme utilisateur.

Le système d'exploitation, organise toutes les fonctions et procédures dans la CPU qui ne sont pas liées à une tâche d'automatisation spécifique. Il gère le déroulement du démarrage à chaud et du redémarrage, l'actualisation de la mémoire image des entrées et l'émission de la mémoire image des sorties, l'appel du programme utilisateur, la gestion des zones de mémoire l'enregistrement des alarmes et l'appel des OB d'alarme...

Le programme utilisateur contient toutes les fonctions nécessaires au traitement des tâches d'automatisation spécifique. Ce programme doit être créé et chargé dans la CPU par l'utilisateur. Il détermine les conditions pour le démarrage à chaud et le redémarrage de la CPU (par exemple, initialiser des signaux), il traite les données du processus (par exemple, combiner des signaux binaires, lire et exploiter des valeurs analogiques), il doit réagir aux alarmes et traiter les perturbations dans le déroulement normal du programme.

Le STEP7 permet de structurer le programme utilisateur en le subdivisant en différentes parties autonomes ou dépendantes. Ceci permet d'écrire des programmes importants mais clairs, simples à tester et à modifier.

• Blocs d'organisation

Les blocs d'organisation (OB) constituent l'interface entre le système d'exploitation et le programme utilisateur. Ils sont appelés par le système d'exploitation et gèrent le traitement du programme cyclique et des programmes déclenchés par alarmes, ainsi que le comportement à la mise en route de l'automate programmable et le traitement des erreurs.

Les blocs d'organisation définissent l'ordre (événements de déclenchement) dans lequel les différentes parties du programme sont traitées. L'exécution d'un OB peut être interrompue par l'appel d'un autre OB (les OB de priorité plus élevée interrompent les OB de priorité plus faible). Les blocs d'organisations les plus prioritaires sont ceux de la mise en route (OB100,

OB101 et OB102) et le moins prioritaire est le cycle en arrière-plan (OB90).

On appelle alarmes les événements qui déclenchent l'appel d'un OB donné. Le type d'alarme définit la classe de priorité de celle-ci.

• Fonctions et blocs fonctionnels

On peut programmer chaque bloc d'organisation en tant que programme structuré en créant des fonctions (FC) et des blocs fonctionnels (FB).

- Les blocs fonctionnels (FB) sont des blocs de code associés à des blocs de données d'instance, dans lesquels sont sauvegardés les paramètres effectifs et les données statiques des blocs fonctionnels.

- Les fonctions (FC) sont des blocs de code sans rémanence, c'est-à-dire qu'ils ne sont pas associés à des blocs de données, les paramètres effectifs ne sont pas sauvegardés automatiquement.

De plus il existe les blocs fonctionnels système (SFB) et les fonctions système (SFC), qui sont des fonctions préprogrammées. Ils peuvent être appelés à partir du programme utilisateur. On trouve des fonctions système pour la copie de blocs de données, le contrôle du programme utilisateur, la gestion des alarmes horaires et temporisées...

• Bloc de données

Les blocs de données (DB) servent à l'enregistrement de données utilisateur. Les blocs de données globales servent à l'enregistrement de données qui peuvent être utilisées par tous les autres blocs. Les blocs de données d'instances sont affectés à des blocs fonctionnels. Les différents blocs cités ci-dessus peuvent être édités avec l'application « CONT LIST LOG ».

La programmation des blocs de codes peut se faire à l'aide de trois applications :

1. CONT LIST LOG : elle permet de programmer des blocs d'organisations OB, des blocs fonctionnels FB et des fonctions FC.
2. GRAPH : elle permet de programmer des blocs fonctionnels FB.
3. SCL : elle permet de créer des sources de code. Une source de code est un fichier texte, qui contient une suite d'instructions, une fois compilé il peut être transféré dans la CPU. On peut

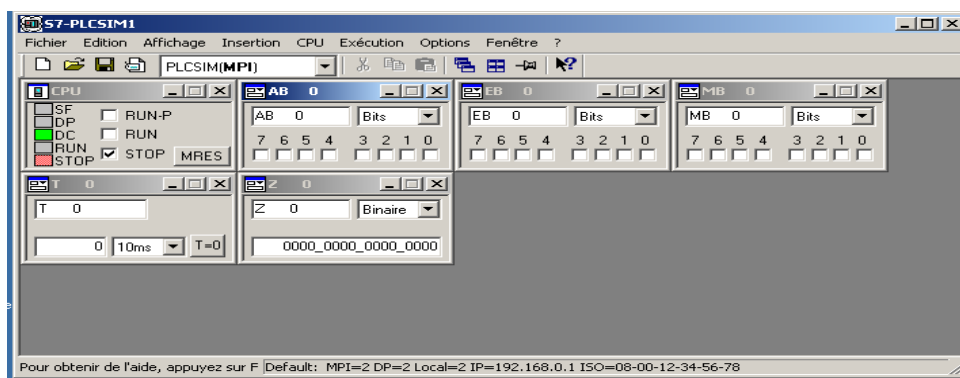
trouver dans un même fichier source tout le programme utilisateur, c'est-à-dire les blocs d'organisations, les blocs fonctionnels et les fonctions.

e- Simulation de modules Avec le logiciel PLCSIM :

S7-PLCSIM dispose d'une interface simple permettant de visualiser et de forcer les différents paramètres utilisés par le programme (comme, par exemple, d'activer ou de désactiver des entrées).

En outre, S7-PLCSIM possède les fonctions suivantes :

- On peut créer des « fenêtres » dans lesquelles on a la possibilité d'accéder aux zones de mémoire d'entrée et de sortie, aux accumulateurs ainsi qu'aux registres de la CPU de simulation. On peut également accéder à la mémoire par adressage symbolique (il faut juste charger la table des mnémoniques dans 'options', puis sur 'outils' 'insérer mnémoniques').
- On peut sélectionner l'exécution automatique des temporisations ou encore les définir et les réinitialiser manuellement.
- On a la possibilité de changer l'état de fonctionnement de la CPU (STOP, RUN et RUNP) comme pour une CPU réelle. De plus, on dispose d'une fonction de pause qui permet d'interrompre momentanément la CPU, sans affecter l'état du programme.



FigureII.7: logiciel de simulation PLC-SIM

f- Chargement du programme dans la CPU :

Une fois la configuration, le paramétrage et la création du programme terminés, on peut transférer le programme utilisateur dans le système cible. La CPU contient déjà le système d'exploitation.

PARTIE B : WinCC, progiciel de conception des interfaces HMI

II.B.1 Présentation de WinCC :

II.B.1.1 Description générale :

Un système IHM constitue l'interface entre l'homme (opérateur) et le processus (machine /Installation). Le contrôle proprement dit du processus est assuré par le système d'automatisation. Il existe par conséquent une interface entre l'opérateur et WinCC flexible (sur le pupitre opérateur) et une interface entre WinCC flexible et le système d'automatisation. Un système IHM se charge des tâches suivantes :

- Représentation du processus :
Le processus est représenté sur le pupitre opérateur. Lorsqu'un état du processus évolue p. ex., l'affichage du pupitre opérateur est mis à jour.
- Commande du processus :
L'opérateur peut commander le processus via l'interface utilisateur graphique. Il peut p. ex. définir une valeur de consigne pour un automate ou démarrer un moteur.
- Vue des alarmes :
Lorsque surviennent des états critiques dans le processus, une alarme est immédiatement déclenchée, p. ex. lorsqu'une valeur limite est franchie.
- Archivage de valeurs processus et d'alarmes :
Les alarmes et valeurs processus peuvent être archivées par le système IHM. On peut ainsi documenter la marche du processus et accéder ultérieurement aux données de la production écoulée.
- Documentation de valeurs processus et d'alarmes :
Les alarmes et valeurs processus peuvent être éditées par le système IHM sous forme de journal. On peut ainsi consulter les données de production à la fin d'une équipe p. ex.
- Gestion des paramètres de processus et de machine :
Les paramètres du processus et des machines peuvent être enregistrés au sein du système IHM dans des recettes. Ces paramètres sont alors transférables en une seule opération sur l'automate pour démarrer la production d'une variante du produit p. ex.

II.B.1.2 Fonctionnement du WinCC :

WinCC est un système modulaire. Ses éléments de base sont le logiciel de configuration (CS) et le logiciel Runtime (RT) :

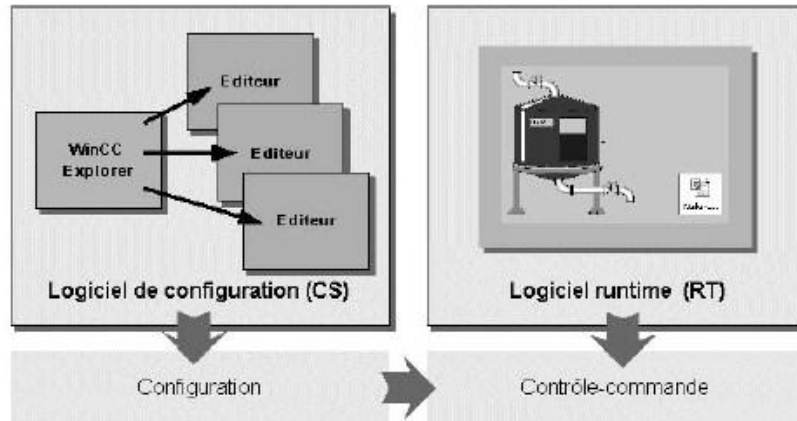


Figure II.8 : Structure interne du WinCC

a. Logiciel de configuration (CS) :

Après démarrage de WinCC, l'écran affiche l'explorateur WinCC Explorer. WinCC Explorer est le noyau du logiciel de configuration. WinCC Explorer affiche la structure complète du projet et permet de gérer le projet. La configuration s'effectue à l'aide d'éditeurs spécifiques que vous pouvez ouvrir à partir de WinCC Explorer. Chaque éditeur permet de configurer un sous-système particulier de WinCC.

Les principaux sous-systèmes de WinCC sont :

- Le système graphique (Graphics Designer)
- Le système de signalisation (Alarm Logging)
- Le système d'archivage (Tag Logging)
- Le système de journalisation (Report Designer)
- Le gestionnaire des utilisateurs (User Administrator)
- La communication – elle se configure directement sous WinCC Explorer
- Toutes les données de configuration sont enregistrées dans la base de données CS.

b. Logiciel Runtime :

Le logiciel runtime permet à l'opérateur d'assurer la conduite du process. Les tâches incombant au logiciel runtime sont les suivantes :

- Lecture des données enregistrées dans la base de données CS
- Affichage des vues à l'écran
- Communication avec les automates programmables
- Archivage des données actuelles de runtime, par exemple des valeurs de process et événements de signalisation

Conduite du process, par exemple spécification de consignes, mise en marche/arrêt.

II.B.1.3 Eléments de l'interface utilisateur de WinCC flexible :

L'environnement de travail de WinCC flexible se compose de plusieurs éléments. Certains de ces éléments sont liés à des éditeurs particuliers et uniquement visibles lorsque cet éditeur est activé.

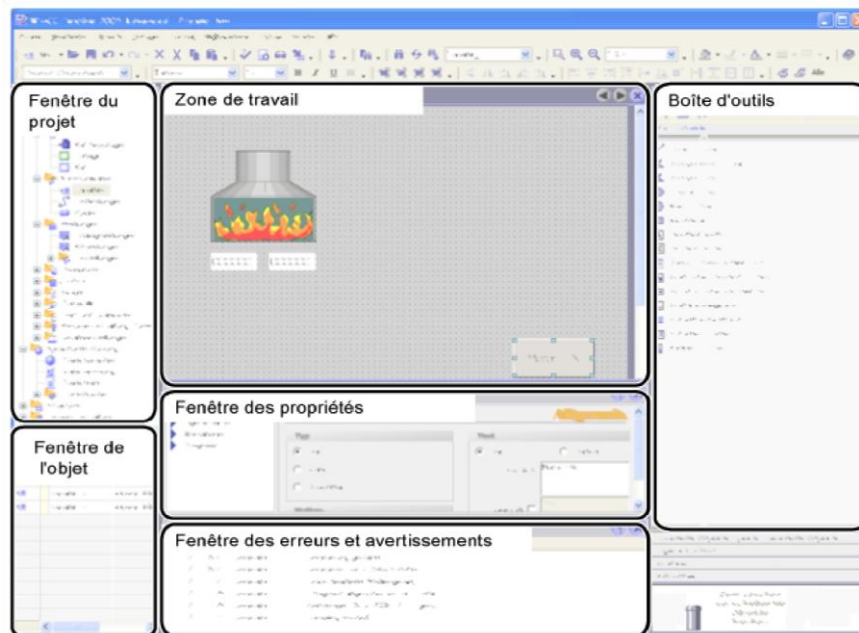


Figure II.9 : Eléments de l'interface WinCC

WinCC flexible se compose des éléments suivants : [5]

a- Menus et barres d'outils

Nous trouverons dans les menus et barres d'outils toutes les fonctions dont nous avons besoin pour la configuration de notre pupitre opérateur. Lorsqu'un éditeur est actif, les commandes de menu ou barres d'outils correspondantes sont visibles. Les menus suivants sont disponibles sous WinCC flexible :

"Projet"	Contient des commandes de gestion de projets.
"Edition"	Contient des commandes servant à utiliser le presse-papiers ainsi que des fonctions de recherche.
"Affichage"	Contient des commandes permettant d'ouvrir et de fermer des éléments ainsi que des paramètres des fonctions zoom et plans. Un élément fermé peut être rouvert via le menu "Affichage".
"Insertion"	Contient des commandes pour l'insertion de nouveaux objets.
"Format"	Contient des commandes servant à disposer et à formater des objets de vue.
"Blocs d'affichage"	Contient des commandes servant à créer et éditer des blocs d'affichage.
"Outils"	Contient, entre autres, des commandes servant à changer de langue d'interface et à configurer les paramètres de base de WinCC flexible.
"Script"	Contient des commandes permettant de synchroniser et de vérifier la syntaxe de scripts.
"Fenêtre"	Contient des commandes de gestion de plusieurs vues de la zone de travail, permettant p. ex. de changer de vue.
"Aide"	Contient des commandes d'accès aux fonctions d'aide.

Tableau II.2: Description des éléments de Menu Sous WinCC flexible

b- Zone de travail

Dans la zone de travail, nous éditons les données de projet soit sous forme de tableau, ce qui est le cas des variables p. ex., soit sous forme graphique, ce qui est le cas des vues de process p. ex. Chaque éditeur ouvert est représenté dans la zone de travail dans un onglet particulier.

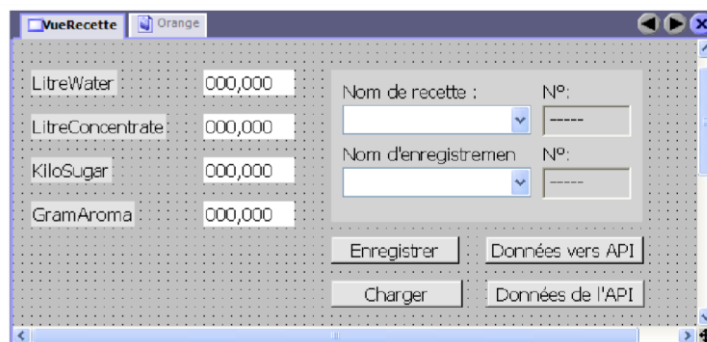


Figure II.10 : La forme graphique d'une Vue.

c- Fenêtre de projet

La fenêtre du projet est le poste central de traitement du projet. Tous les éléments et tous les éditeurs disponibles d'un projet sont affichés sous forme d'arborescence dans la fenêtre du projet.

La fenêtre de projet sert à créer des objets et à les ouvrir pour les éditer. Nous pouvons créer des dossiers pour structurer les objets de notre projet. Nous pouvons ouvrir pour chaque objet un menu

contextuel qui regroupe les principales commandes et nous pouvons aussi accéder aux paramètres du pupitre, à la localisation et à la gestion de versions.

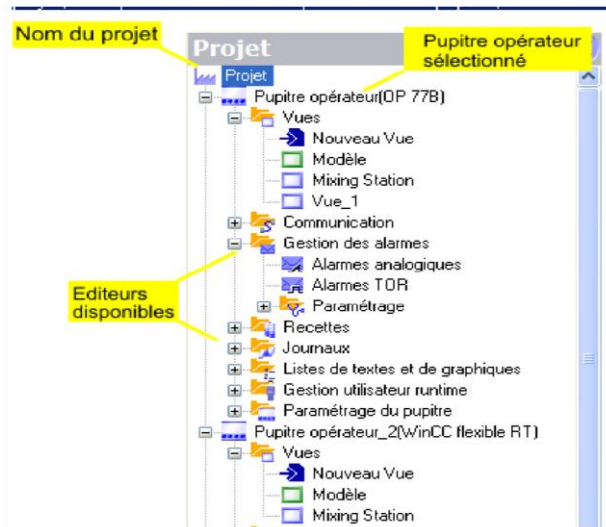


Figure II.11 : fenêtre de projet.

d- Fenêtre des propriétés

La fenêtre des propriétés permet de modifier les propriétés d'un objet sélectionné dans la zone de travail. Le contenu de la fenêtre des propriétés dépend de l'objet sélectionné.

La fenêtre des propriétés affiche les propriétés de l'objet sélectionné classées par catégories. Aussitôt que nous quittons une zone de saisie, les modifications de valeurs effectuées sont actives.

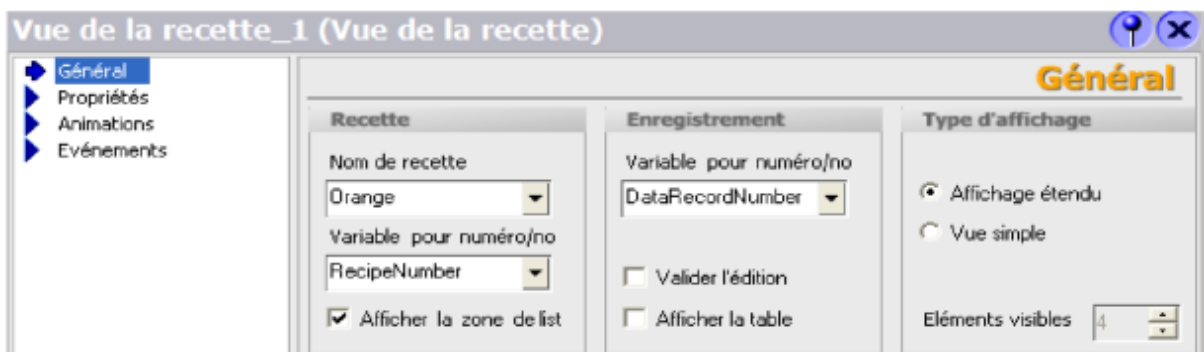


Figure II.12 : Fenêtre des propriétés

e- Bibliothèque

La bibliothèque fait partie de la fenêtre d'outils, elle est un lieu central d'enregistrement des objets fréquemment utilisés. Sous WinCC flexible, on distingue la bibliothèque globale de la bibliothèque spécifique au projet :

- Bibliothèque globale

La bibliothèque globale n'est pas enregistrée avec le projet dans la base de données mais sous forme de fichier. Le fichier enregistré est stocké par défaut dans le répertoire d'installation de WinCC flexible. La bibliothèque globale est disponible pour tous les projets.

- Bibliothèque spécifique projet

La bibliothèque de projet qui est enregistrée avec les données de projet dans la base de données, est uniquement disponible dans le projet où elle a été créée.

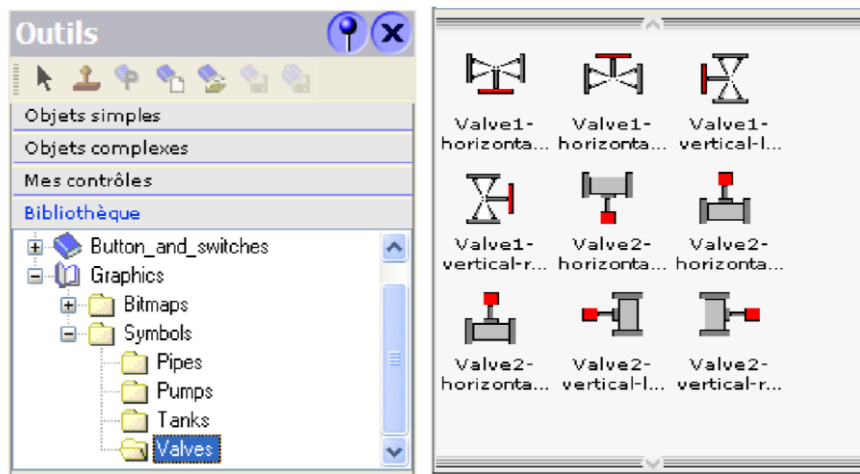


Figure II.13: Élément de la bibliothèque

f- Fenêtre des erreurs et avertissements

La fenêtre des erreurs et avertissements affiche les événements système générés p. ex. lors du test d'un projet.

Dans la fenêtre des erreurs et avertissements, les événements système sont affichés par défaut dans leur ordre d'apparition. Les catégories désignent respectivement le module WinCC flexible qui a généré un événement système. Les alarmes système de la catégorie "Compilateur" sont p. ex. générées durant le contrôle de cohérence.

La fenêtre des erreurs et avertissements affiche tous les événements système se rapportant à la dernière action. A chaque nouvelle action, tous les événements système précédents sont écrasés.



Figure II.14: Fenêtre des erreurs et avertissements.

g- Fenêtre des objets

Lorsque nous sélectionnons des dossiers ou des éditeurs dans la fenêtre de projet, leur contenu s'affiche dans la fenêtre des objets.

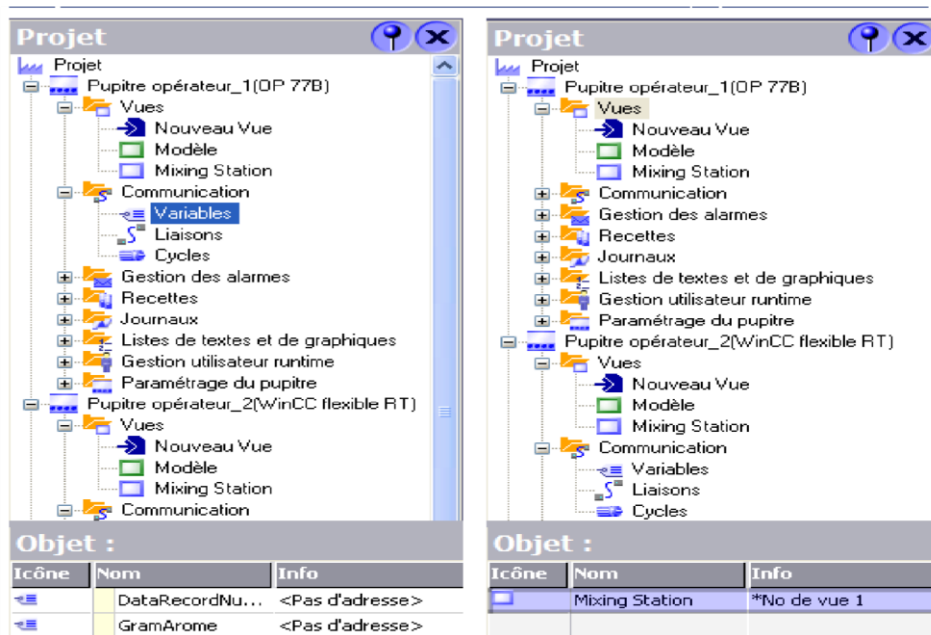


Figure II.15 : Fenêtre des objets

II.B.2.Etapes de conception d'un projet sous WinCC :

Nous allons aborder dans cette partie les différentes étapes pour aboutir à un projet et ceci en expliquant chaque étape.

a. Création d'un projet :

Après avoir lancé le SIMATIC WinCC Explorer, on clique sur : 'Fichier → nouveau' ; Une fenêtre de dialogue affiche alors les différents types de pupitre à choisir :

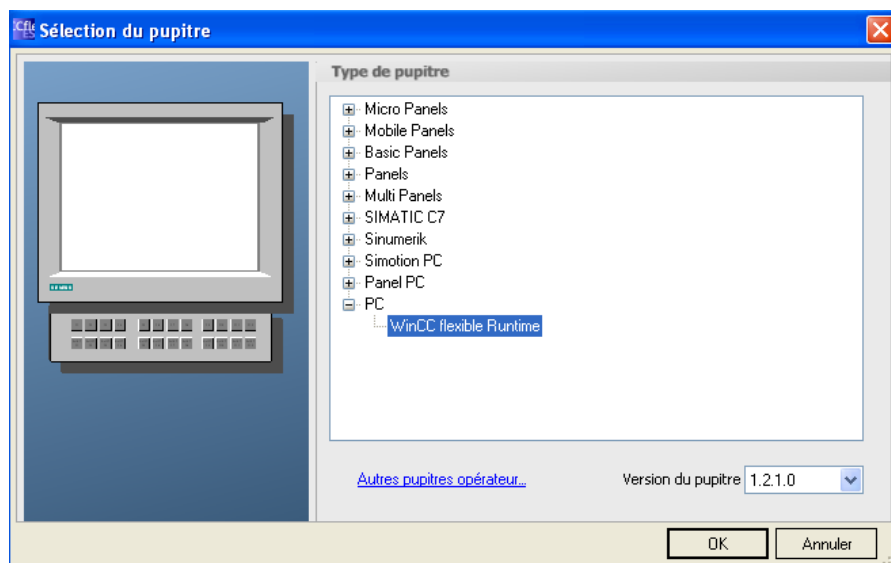


Figure II.16 : Fenêtre de sélection du pupitre

On choisira pour notre projet 'WinCC flexible Runtime' et avant de commencer il faut d'abord configurer la résolution de l'écran pour assurer une bonne supervision.

Dans la fenêtre de projet, on clique sur paramètre de pupitre et on coche sur 'mode plein écran' et on choisit une résolution d'écran correspondante à notre résolution d'écran du système.

- WinCC flexible Runtime

WinCC flexible Runtime est un logiciel performant et facile à utiliser pour la visualisation du processus des projets créés avec le logiciel de configuration WinCC flexible Advanced.

WinCC flexible Runtime est conçu pour la visualisation et l'utilisation de machines et de petites installations. Le logiciel Runtime se distingue par son interface utilisateur entièrement graphique basée sur la technique des fenêtres. Il permet grâce à des temps de réaction rapides une conduite de processus sûre, ainsi qu'une collecte sûre des données. [5]

b- Création d'une vue de process :

Dans WinCC flexible, on crée des vues pour le contrôle-commande des machines et d'installations. Pour créer des vues, vous disposez d'objets prédéfinis permettant de représenter notre installation, d'afficher des procédures et de définir des valeurs de processus.

(Dans la fenêtre de projets : 'vues' → 'ajouter un vue')



Figure II.17 : Une vue principale

Une vue peut être composée d'éléments statiques et d'éléments dynamiques.

- Les éléments statiques : ne changent pas au runtime, p. ex. le texte et le graphique.
- Les éléments dynamiques varient en fonction de la procédure, soit à partir de la mémoire de l'automate programmable soit à du pupitre opérateur, sous forme d'affichages alphanumériques, de courbes et de bargraphes. Même aussi les champs de saisie du pupitre opérateur font également partie des objets dynamiques.

c- Définition des variables:

Les variables utilisées dans WinCC représentent soit des valeurs réelles, comme par exemple le taux de remplissage d'un réservoir d'eau, soit des valeurs internes, qui sont calculées où bien simulées en interne par WinCC :

- Variables externes :

Les variables externes sont des emplacements en mémoire d'un API ou d'un matériel semblable. Ainsi le niveau de remplissage du réservoir d'eau est relevé par un capteur de niveau et enregistré dans l'API. Le taux de remplissage est communiqué à WinCC par le canal de communication.

- Variables internes :

Les variables internes sont des emplacements en mémoire de WinCC qui assurent les mêmes fonctionnalités qu'un API. Elles peuvent être calculées et modifiées en interne par WinCC.

Les groupes de variables servent à structurer les variables. Toutes les variables peuvent, pour plus de clarté, être rangées dans des groupes de variables.

Pour définir notre variable, on clique dans la fenêtre de projet → communication → variable et on choisit la liaison si le variable est externe sinon on le laisse un variable interne.

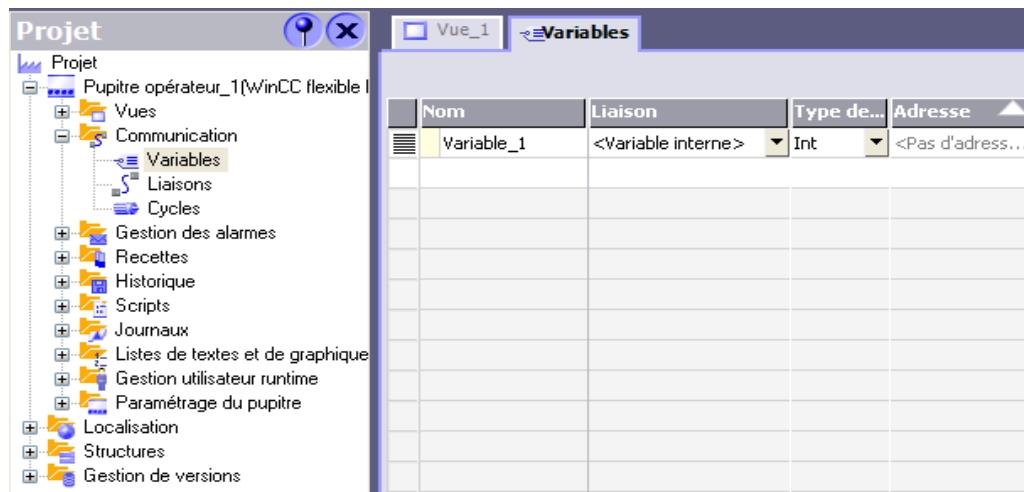


Figure II.18 : Fenêtre des variables

d- Simulation de la vue de process :

Le logiciel de configuration WinCC flexible est fourni avec un simulateur qui permet de tester le projet sans automate. Le simulateur est une application en propre. Il nous permet de contrôler le bon fonctionnement des vues, objets de vue et messages configurés.

Pour simuler le projet terminé, plusieurs possibilités s'offrent :

- a- Simulation avec raccordement à l'automate

Nous pouvons simuler notre projet en l'exécutant directement dans runtime. Mais dans ce cas, les variables et les pointeurs de zone ne peuvent fonctionner que si notre PC de configuration est raccordé à un automate approprié. Si notre ordinateur est raccordé à un automate, Runtime nous permet de réaliser une simulation authentique du pupitre opérateur configuré.

b- Simulation sans raccordement à l'automate

A l'aide du programme de simulation installé en même temps que WinCC flexible Runtime nous pouvons simuler le projet, y compris les variables et les pointeurs de zone sans connexion à un automate. Nous saisissons les paramètres des pointeurs de zone et des variables dans un tableau de simulation qui est lu par WinCC flexible Runtime pendant la simulation.

c- Simulation en fonctionnement intégré

Si nous configurons en mode intégré dans STEP 7, nous pouvons simuler un raccordement à un automate via PLCSIM.

Pour effectuer la simulation avec WinCC flexible Runtime, nous procédons comme suit:

1. on compile notre projet, un fichier génère automatiquement portant l'extension *. fwx sous le même répertoire du projet.
2. on sélectionne dans le menu "Projet" la commande "Générateur > Démarrer runtime". Sinon, on clique dans la barre d'outils "Générateur" sur l'icône.

II.B.3. Configuration des alarmes : [6]

Dans cette partie nous nous intéressons à la procédure de création d'un système d'alarmes pour le Runtime. Le système d'alarmes informe l'opérateur des états de fonctionnement ou pannes du process et il est créé à l'aide de l'éditeur Alarm Loggin.

Les alarmes configurées dans Alarm Logging sont sorties dans Runtime lorsque l'événement correspondant se produit, par exemple un défaut ou un dépassement de seuil.

WinCC flexible offre les options ci-dessous pour afficher des messages sur un pupitre opérateur :

- **Vue des Alarmes :**
Une vue des alarmes est configurée pour une vue déterminée. Selon la taille configurée, il peut afficher plusieurs alarmes simultanément. Vous pouvez configurer plusieurs affichages d'alarmes pour des classes d'alarmes différentes et dans des vues différentes.
- **Ligne des Alarmes:**
La vue des alarmes peut être configuré de manière à ne comporter qu'une seule ligne au Runtime et c'est la dernière alarme qui sera affiché ("Propriétés > Représentation > Mode > Comme ligne d'alarme").
- **Fenêtre des Alarmes:**
Une vue des alarmes est configurée pour une vue déterminée. Selon la taille configurée, il peut afficher plusieurs alarmes simultanément. Vous pouvez configurer plusieurs affichages d'alarmes pour des classes d'alarmes différentes et dans des vues différentes.
- **Indicateur de présence d'Alarmes :**
L'indicateur d'alarme est un symbole graphique configurable qui s'affiche à l'écran quand une alarme apparaît et qui doivent être acquittées selon le paramétrage réalisé .L'indicateur de message d'alarmes ne peut qu'être configuré qu'en tant qu'icône graphique.

Vous trouverez ci-après des astuces pour la création et l'édition des Vues/Fenêtres d'alarmes.

Déplacement des colonnes de la Vue/Fenêtre d'alarme :

Sélectionner votre Vue/Fenêtre d'alarme puis faites un Clic droit .Dans le menu déroulant, choisissez "Editer". Vous pouvez alors avec le bouton gauche de la souris déplacer et arranger individuellement les titres des colonnes.

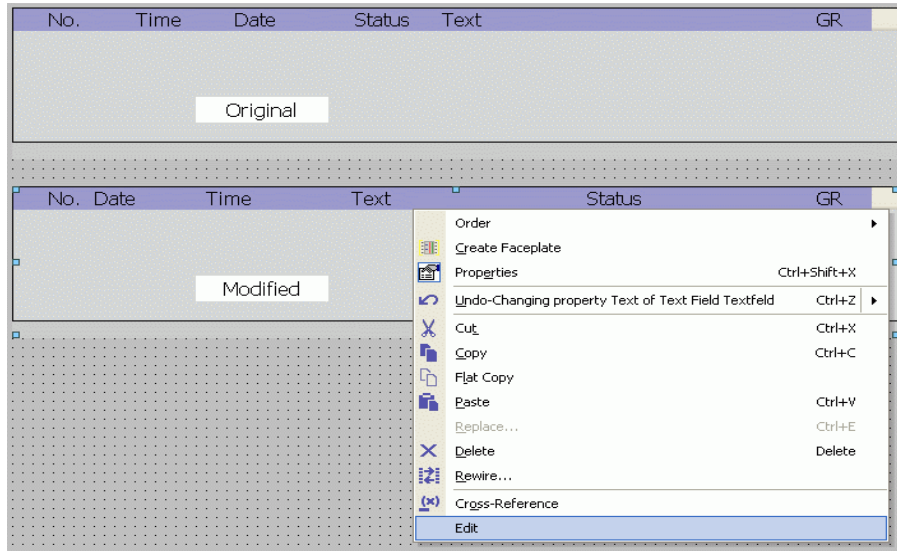


Figure II.19 : Fenêtre d’alarme

Création de votre propre classe d'alarme :

Vous pouvez ajouter de nouvelles classes d'alarmes à celles des classes déjà existantes (Erreurs, Avertissements, Système, Événements de diagnostic). Il est ainsi possible d'affecter une classe d'alarme séparée pour chacun de vos sous-systèmes. Vous pouvez alors affecter la classe d'alarme en conséquence dans votre Vue/Fenêtre d'alarme.

Il n'y aura alors que les alarmes de votre sous-système sélectionné qui seront affichées.

La "Classe d'alarme" se trouve dans l'arborescence du projet "Gestion des alarmes > Paramétrage > Classe d'alarmes". L'image ci-après vous montre la modification d'une classe d'alarme dans la vue des Alarmes.

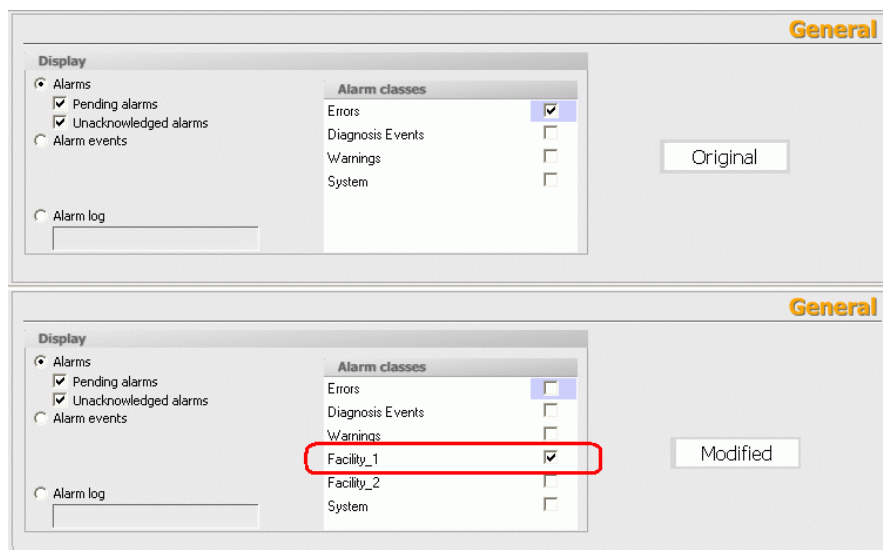


Figure II.20 : Fenêtre classe d’alarme

Boutons d'édition:

Avec le menu "Propriétés > Paramétrage " de la Vue/Fenêtre des Alarmes, vous pouvez affecter les boutons pour l'affichage (Texte d'Info, Acquiescement, Editer). Vous ne pouvez pas changer les icônes de ces boutons. Si vous souhaitez utiliser vos propres Icônes/Textes, vous devez créer vos propres Boutons sur lesquels vous les Symboles / Textes de votre choix. Les fonctions sont disponibles dans le menu des propriétés des boutons ("Propriétés > Evénements > Appuyer > Clavier pour objets de vues > Vue alarme...").

Configuration de la couleur des classes d'alarmes :

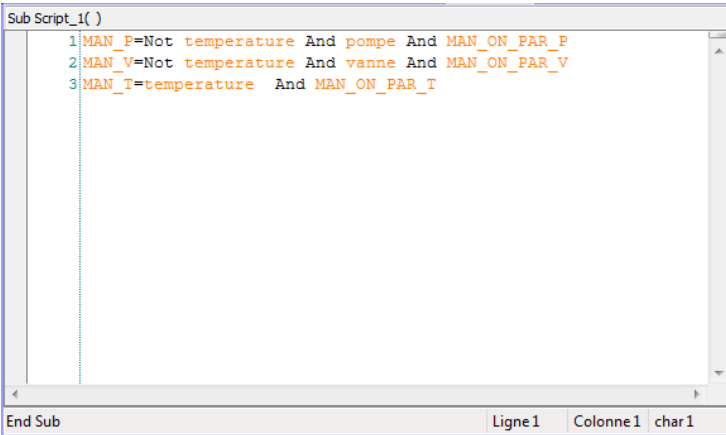
Afin de mieux différencier les états des alarmes (arrivé, parti, acquitté) vous pouvez configurer la couleur des classes d'alarmes, vous devez valider "Utiliser Couleur de classes d'alarmes" dans la fenêtre de Projet sous l'arborescence "Gestion des alarmes > Paramétrage > Paramétrage des alarmes > Général".

Variables configurées dans les textes des messages d'alarmes:

Si vous avez configuré une Variable dans un texte de messages pour par exemple afficher la valeur en cours au moment de l'erreur, cette valeur est dite "statique". Cela signifie que la valeur de la variable est affichée avec la valeur que la variable avait au moment où le message est apparu .Cette valeur n'est alors plus mise à jour dans la Vue/Fenêtre des Alarmes.

II.B.4. Création de fonctions et d'actions (Global Script) :

Le Global Script est l'éditeur dans lequel on peut créer des fonctions et des actions dans le but de dynamiser l'environnement WinCC Runtime.



```

Sub Script_1( )
1 MAN_P=Not temperature And pompe And MAN_ON_PAR_P
2 MAN_V=Not temperature And vanne And MAN_ON_PAR_V
3 MAN_T=temperature And MAN_ON_PAR_T
End Sub

```

The screenshot shows a text editor window titled "Sub Script_1()". It contains three lines of VBS code: "1 MAN_P=Not temperature And pompe And MAN_ON_PAR_P", "2 MAN_V=Not temperature And vanne And MAN_ON_PAR_V", and "3 MAN_T=temperature And MAN_ON_PAR_T". The status bar at the bottom indicates "End Sub", "Ligne 1", "Colonne 1", and "char 1".

Figure II.21: Global Script.

VBScript (VBS) nous permet d'accéder en Runtime aux variables et aux objets du système graphique Runtime, mais aussi d'exécuter des actions ne dépendant pas des vues:

- Variables: On peut lire et écrire des valeurs variables pour par exemple proposer des valeurs variables pour l'automate, et ce par simple clic de souris sur un bouton.

- Objets: On peut dynamiser les propriétés de l'objet avec des actions et déclencher des actions via les événements pour objets.
- Actions ne dépendant pas de la vue: On peut déclencher des actions indépendamment des vues, de façon cyclique ou par le biais de valeurs variables, par exemple la transmission quotidienne de valeurs dans un tableau Excel.

VBS dans WinCC nous permet d'utiliser des procédures, des modules et des actions permettant de dynamiser l'environnement Runtime:

- Procédures: Une procédure correspond à une fonction en C. On inscrit dans les procédures le code qu'on veut utiliser en plusieurs endroits de la configuration.
- Modules: Les modules nous permettent de regrouper les procédures en unités significatives. On crée par exemple des modules pour des procédures qui sont utilisées dans une vue déterminée ou qui font partie d'une rubrique particulière, par exemple les fonctions auxiliaires mathématiques ou les fonctions d'accès à la banque de données.
- Actions: Les actions sont toujours déclenchées par un Trigger. On configure les actions pour des propriétés d'objets graphiques, pour des événements qui interviennent pour un objet graphique ou bien globalement dans le projet. Dans les actions, on peut appeler sous forme de procédures un code utilisé plusieurs fois.

Conclusion :

Munis d'un logiciel très performant, les automates programmables de la série S7-300/S7-400 forment des unités de traitement et de commande d'une grande flexibilité. Dans ce chapitre nous avons pu voir que le STEP7 qui est un logiciel très performant grâce à sa flexibilité, la diversité de ses langages de programmation, la possibilité de la simulation et le traitement des variables de type réel. Nous avons vu aussi que le WinCC est un logiciel indispensable pour la supervision des processus ainsi que l'exploitation des données acquises à partir de ces capteurs et de la CPU.

CHAPITRE III

Description et identification des différents systèmes De la station Process Trainer (CE117).

Introduction :

La station CE117 nous offre une variété de systèmes qu'on peut commander par des différentes méthodes. Dans ce chapitre on va voir la description et l'identification de ces systèmes.

PARTIE A : Description de la station TecQuipment Process Trainer CE117

III.A.1. Description de la station TecQuipment Process Trainer CE117 :

La station CE117 Process Trainer est une appareil éducative destiné à l'étude et le contrôle de plusieurs systèmes : contrôle de niveau, pression, température et en fin de débit, qui sont des systèmes rencontré très souvent dans le monde industriel. La station est composée de trois parties commandées par un module de contrôle, chaque une regroupe plusieurs équipements nécessaires pour le contrôle des différentes grandeurs:

- 1- Le circuit fluide procédé : réservoir, pompe P2, vanne By-pass, capteur de température avant refroidissement, refroidisseur, capteur de température après refroidissement, vanne proportionnelle, capteur transmetteur de débit.
- 2- Le circuit fluide chaud : cuve de chauffe, capteur de température de fluide de la cuve de chauffe, pompe P1, capteur de débit, capteur de température à l'entrée de la cuve de chauffe.
- 3- La cuve agitée : vanne de mise à l'atmosphère, capteur transmetteur de niveau et de pression de l'air comprimé dans la cuve, agitateur, capteur transmetteur de température de fluide, le serpentin de chauffe de fluide, vanne de drainage.

Les équipements de cette appareil permettent de contrôler le débit, le niveau, la pression et la température non seulement séparément mais aussi en interaction et en cascade.

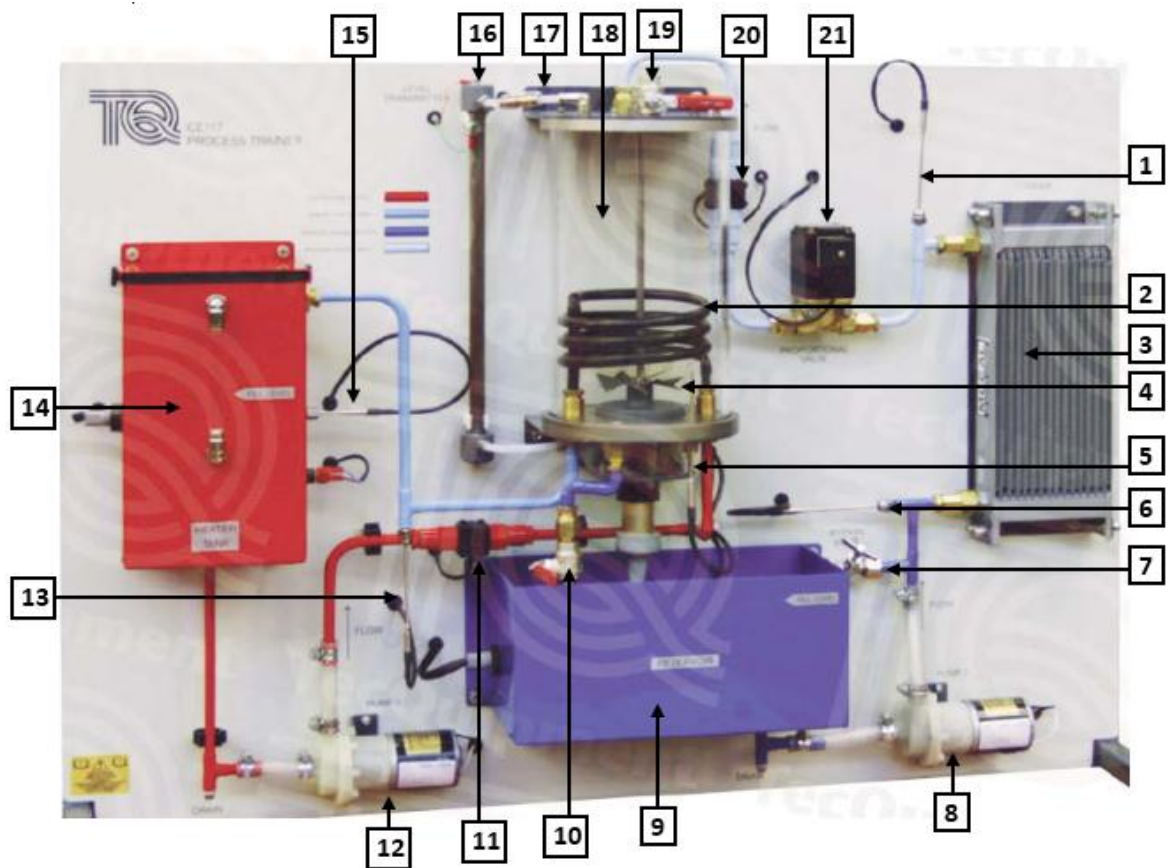


Figure III.1 : la station CE117 Process Trainer

- | | |
|-------------------------------------|--------------------------------------|
| 1 - transmetteur de température TT4 | 12 - pompe 1 |
| 2 - serpentin de chauffe | 13 - transmetteur de température TT2 |
| 3 – refroidisseur | 14 - cuve de chauffe |
| 4 – Agitateur | 15 - transmetteur de température TT1 |
| 5 - transmetteur de température | 16 - transmetteur de niveau LT |
| 6 - transmetteur de température TT3 | 17 - transmetteur de pression PT |
| 7 - vanne By-pass | 18 – cuve agitée |
| 8 – pompe 2 | 19 – vanne vent |
| 9 - réservoir | 20 – transmetteur de débit FT2 |
| 10 – vanne de drainage | 21 – vanne proportionnelle |
| 11 – transmetteur de débit FT1 | |

- **Le module de contrôle :**

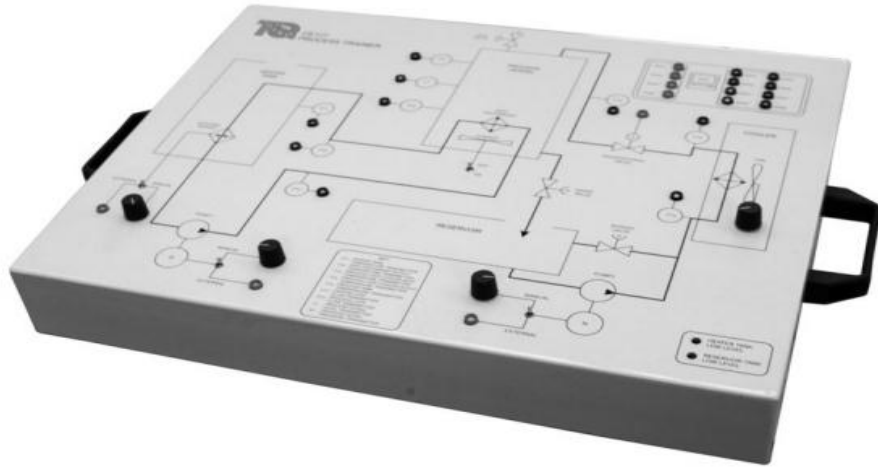


Figure III.2 : Le module de contrôle de la station CE117.

La station CE117 est associée d'un pupitre qui représente un module de contrôle. Ce module regroupe l'ensemble des circuits de câblage des actionneurs. Pour le contrôle en temps réel et l'acquisition des données par le logiciel CE2000 le module de contrôle de TecQuipment Process Trainer (CE117) contient une interface intégrée ADA.

On trouve sur ce module un schéma du procédé ainsi que les connexions des différents capteurs-transmetteurs et actionneurs. L'interface ADA peut recevoir 8 entrées de la part des capteurs, ces entrées de nature analogique seront converties en numérique. L'interface peut également fournir 4 sorties converties en analogique destinées aux actionneurs.

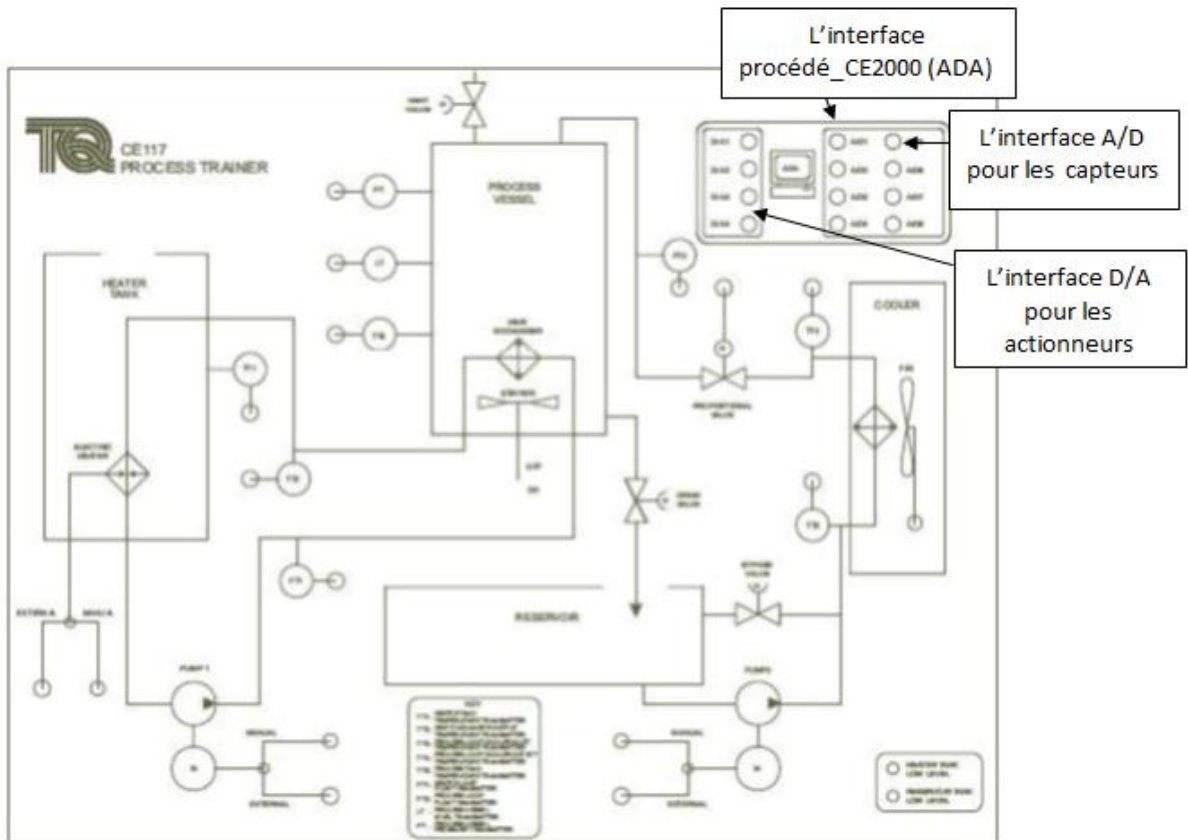


Figure III.3 : Schéma du module de contrôle de la station CE117.

L'interface ADA est considéré comme le seul moyen de communiqué le PC avec la station CE117 en utilisant le logiciel CE2000. Si on utilise un automate programmable on n'aura pas besoins de cette interface, on relie directement les capteurs au module d'entrée analogique et les actionneurs au module de sortie analogique, sauf si l'automate ne dispose pas d'entrées et de sorties suffisantes pour la commande du procédé.

III.A.2. Les caractéristiques techniques des différents éléments de la station CE117 :

Le tableau suivant résume les caractéristiques techniques des différents éléments de la station CE117 :

article		Signal Analogique	Détail de la conversion
PRT Température Transmetteurs (résistance platiniim Thermomètre)	TT1	Sortie 0-10V linéaire	10°C par Volt 0V= 0°C 10V= 100°C
	TT2		
	TT3		
	TT4		
	TT5		
Transmetteur de débit		Sortie 0-10V linéaire	1L/min par Volt 0V=pas de débit
FT1	FT2		
Transmetteur de niveau		Sortie 0-10V non linéaire	0V=réceptant vide 10V=Niveau Maximum
Transmetteur de pression		Sortie 0-10V linéaire	100mbar par volt 0V=0mbar (jauge)
chauffage électrique		Sortie 0-10V	75W par volt 0V=chauffage OFF 10V=750W puissance max(Nominale)
Vanne Proportionnelle		Sortie 0-10V	0V=fermée 10V=ouverte
S			
Pompe 1 Pompe 2		Sortie 0-10V	0V=pas de débit 10V=débit Maximal

Tableau III.1 : les caractéristiques techniques des équipements de la station CE117

III.A.2.1 Les actionneurs :

1- La pompe :

La pompe est utilisé pour délivré un débit d'eau dans notre circuit, et elle est contrôlé par une tension qui va de 0 jusqu'à 10V, mais il faut prendre en compte qu'elle ne fonctionne pas pour des tensions au-dessous de 2V, et elle ne peut délivrer que 5.2 L/min comme débit maximale pour une tension de 10 V.

2- La vanne :

La vanne est contrôlée en ouverture par une tension qui varie entre 0 et 10V.

III.A.2.2 Les capteurs :

1- Le capteur de débit : C'est un capteur linéaire qui donne une image de débit sous forme une tension qui varie entre 0 et 10V (1V correspond à 1L/min). Ce capteur linéaire est très rapide, sa fonction de transfert est donc un gain unitaire.

- 2- Le capteur de pression : C'est un capteur utilisé pour mesurer la pression de l'air comprimé dans la cuve agitée, il peut mesurer jusqu'à 1000 mbar. Comme ce capteur est linéaire et très rapide alors sa fonction de transfert est donnée par un gain de 0.01.
- 3- Le capteur de température : C'est un capteur utilisé pour mesurer la température dans différents endroits de la station, il peut aller jusqu'à 100°C. Comme ce capteur est linéaire et très rapide alors sa fonction de transfert est donnée par un gain de 0.1.
- 4- Le capteur de niveau : C'est un capteur non linéaire de fonction de transfert variable, et comme il est très rapide on peut la considérer comme un gain variable. Ce capteur est linéaire de gain unitaire dans la plage de travail entre 0 - 4 Cm comme montre la figure ci-dessous.

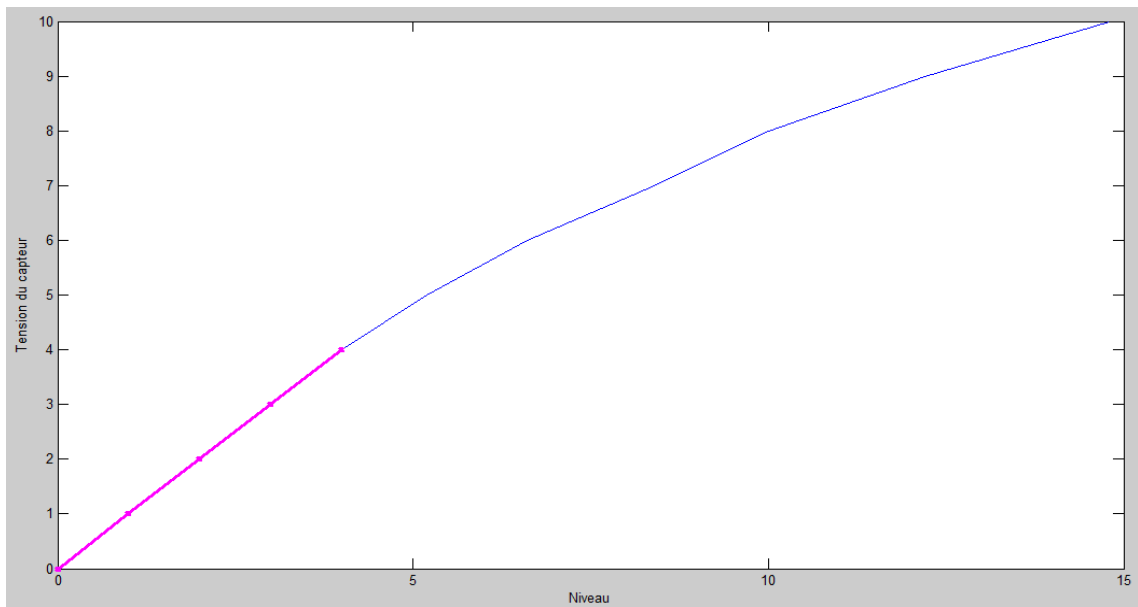


Figure III.4 : La caractéristique statique du capteur de niveau

Le tableau suivant donne la caractéristique du capteur de niveau :

Tension (V)	Niveau (Cm)	Gain Kc
1	1	1
2	2	1
3	3	1
4	4	1
5	5.2	0.96
6	6.6	0.90
7	8.4	0.83
8	10	0.80
9	12.2	0.73
10	14.8	0.67

Tableau III.2 : Tableau qui résume la caractéristique du capteur de niveau

III.A.3. Le logiciel TecQuipment Process Trainer CE2000 :

Le CE2000 est un logiciel de contrôle puissant avec beaucoup de fonctionnalités. Il est fourni en standard avec le contrôleur (CE120), l'interface digitale (CE122) et le formateur de processus (CE117). On utilise les icônes de logiciels et on câble l'ensemble sur l'écran tout comme on trace un système de contrôle sur un morceau de papier.

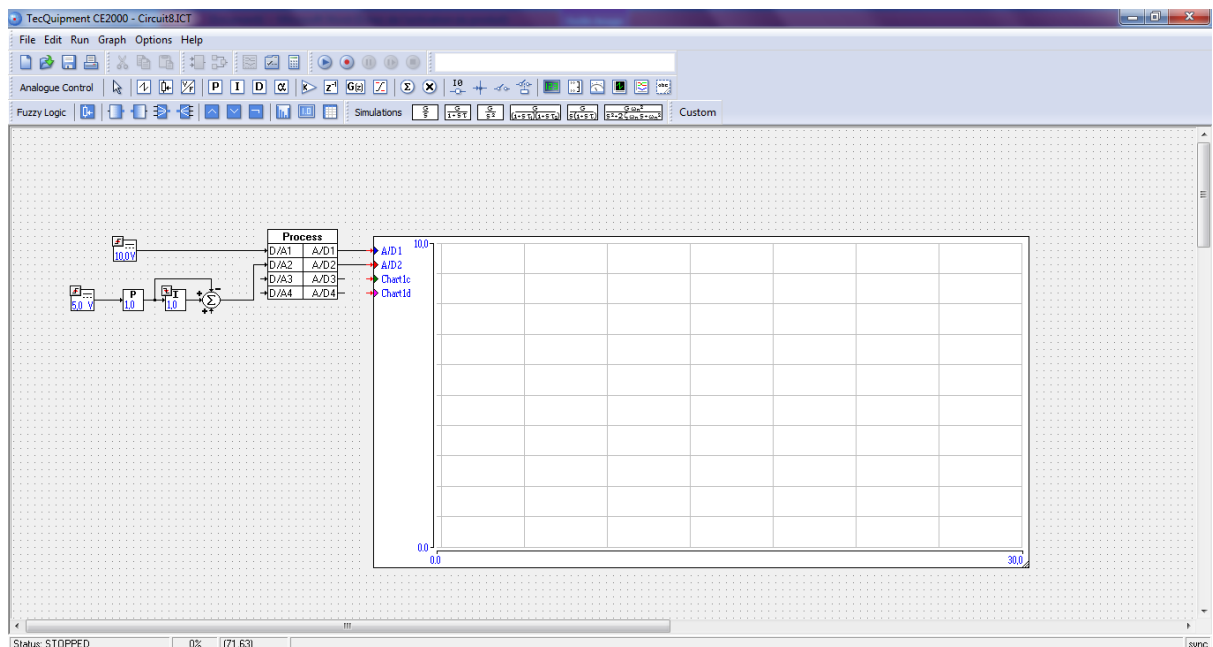


Figure III.5 : L'interface de logiciel CE2000

Les icônes de contrôleurs, générateurs de signaux et tensions sont les parties les plus importantes pour la commande manuelle ou automatique du système. Avec le logiciel CE2000, on peut créer un ou plusieurs types de contrôleurs et simuler les réponses théoriques où réelles sélectionnés dans

TecQuipment Process Trainer (CE117), qui donne une large sélection de systèmes : linéaires et non linéaires, stables et instables, multi-modèles.

Pour lancer la simulation il suffit d'appuyer sur « *Run* » dans la barre d'outils, pour sauvegarder les résultats il faut appuyer sur le bouton *Record* ou bien choisir « *Record data* » dans la liste de « *Run* ».

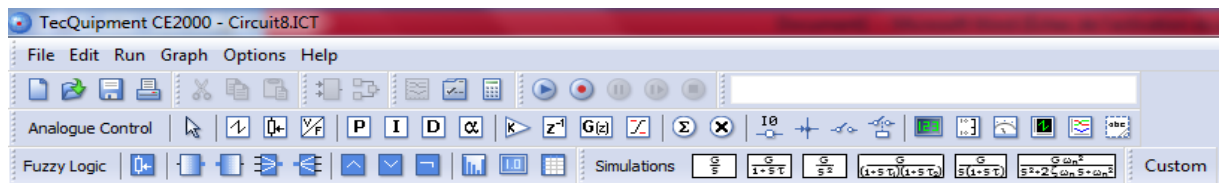


Figure III.6 : La barre d'outils de logiciel CE2000

Le logiciel est capable d'enregistrer les résultats dans un tableau et d'exporter les données pour une utilisation dans d'autres programmes. Pour exporter les fichiers de données il suffit de cliquer sur « *File* » et puis choisir « *Export data file* », le fichier va être enregistré dans un bloc-notes qu'on peut s'en servir après.

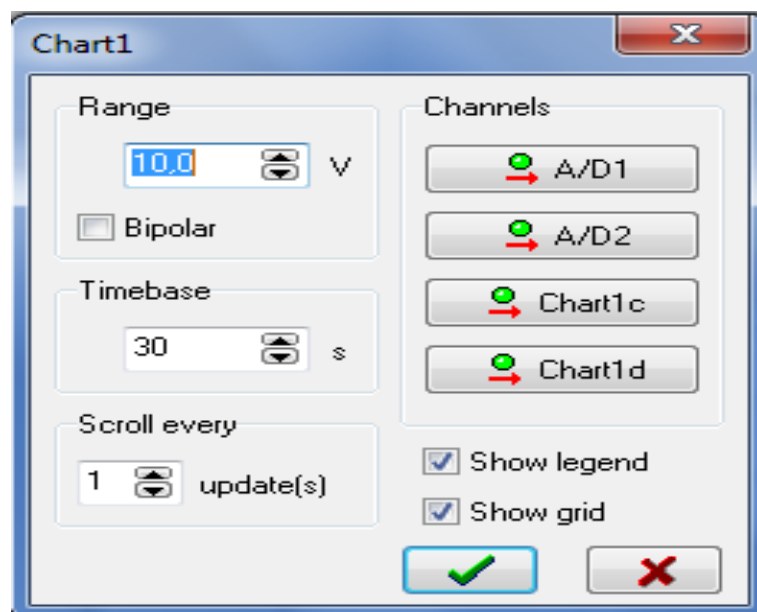


Figure III.7 : la fenêtre de graphe de logiciel CE2000

Pour pouvoir modifier les propriétés du graphe il suffit de maitre la souris sur le graphe et appuyer sur le bouton droit de la souris, puis sur « *properties* » la fenêtre présentée dans la figure précédente va apparaître pour nous permettre de faire les modifications qui convient.

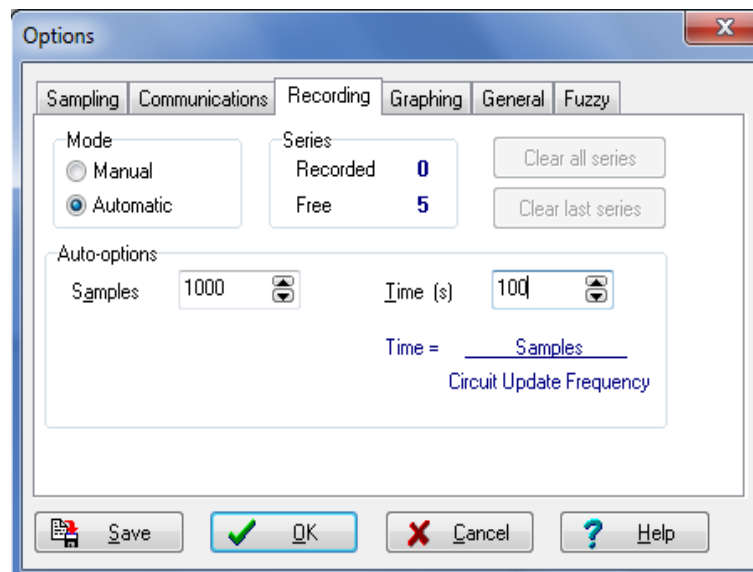


Figure III.8 : fenêtre d'options des graphes enregistrés de logiciel CE2000

Pour pouvoir modifier les propriétés de l'enregistrement, il suffit d'appuyer sur le bouton « *Graph* » dans la barre d'outils, puis sur « *Options* », la fenêtre présentée dans la figure précédente va apparaître pour nous permettre de faire les modifications.

III.A.4. Les contraintes des différents systèmes de la station CE117 :

- 1- La pompe possède une zone morte (si la tension de commande de la pompe descend au-dessous de 2V, elle ne délivre plus de débit).
- 2- Le capteur de niveau est non linéaire avec un gain variable.
- 3- La pompe ne peut délivrer que 5.2 (L/min) comme débit maximal.
- 4- La perturbation sur le débit provoqué par la vanne By-Pass peut atteindre 0.7 (L/min), ce qui va réduire le maximum de la référence de débit à 4.5 (L/min).
- 5- Si la pompe délivre un débit avec une pression inférieure à celle de l'air comprimé dans la cuve, le débit s'annule et la pression se stabilise à sa valeur malgré qu'il y ait une tension à l'entrée de la pompe.
- 6- L'automate programmable S7 314 IFM ne dispose que d'une seule sortie analogique, ce qui n'est pas suffisant pour la commande de notre système.

PARTIE B. Identification

III.B.1.1. le but d'identification

Pour arriver aux objectifs décrits dans le cahier des charges de la régulation d'un procédé, il faut analyser les comportements statique et dynamique de ce procédé seul, ou instrumenté, c'est-à-dire connaître sa fonction de transfert ou son modèle d'état. En effet, le réglage du correcteur à mettre en œuvre dépend essentiellement de la nature de cette fonction de transfert. [7]

III.B.1.2. Les méthodes d'identification

L'identification qui consiste à déterminer la fonction de transfert d'un système peut être effectuée par une mise en équation du système. Pour des systèmes simples, ou qui peuvent être décomposé en éléments simples, cela conduit à un modèle de connaissance.

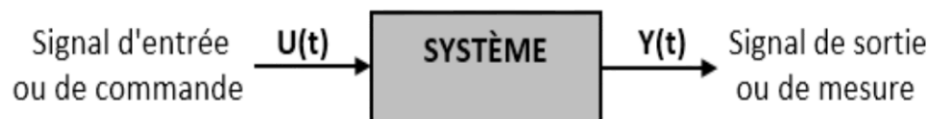
Lorsque le procédé est complexe et sa mise en équation est délicate, voire impossible par manque de connaissance des coefficients mis en jeu, une identification expérimentale est préférable. [7]

Parmi les nombreuses méthodes expérimentales existantes, on s'intéresse dans cette partie à présenter que les méthodes utilisé par la suite.

Identification en boucle ouvert :

a- Méthodologie :

On envoie un signal d'entrée $U(t)$ connu (échelon, impulsion ou rampe) et on enregistre la sortie $Y(t)$ qui est analysé ensuite.



Les trois grandes familles de courbes des mesures les plus rencontrées sont les suivantes :

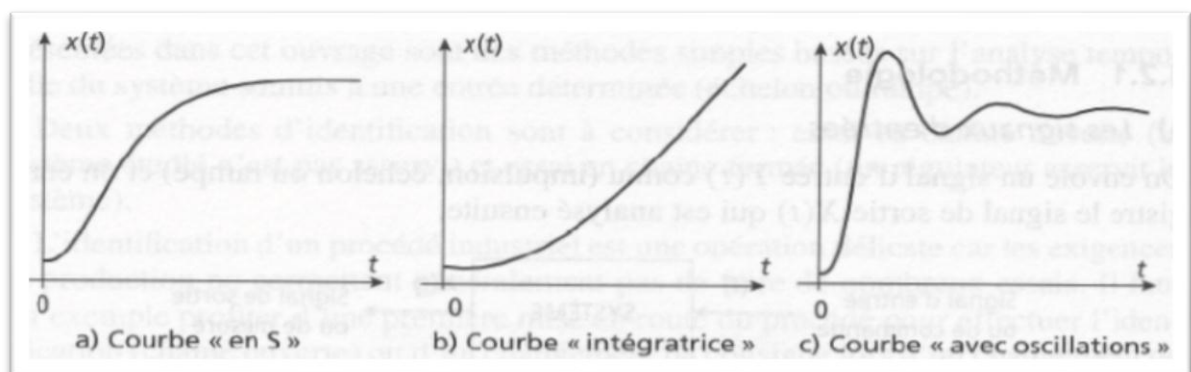


Figure III.9 : Les trois grandes familles de courbes

L'allure de la courbe obtenue suggère la méthode à appliquer pour modéliser le système. La fonction de transfert $H(s)$ obtenue est le rapport de la fonction de signal de sortie $Y(s)$ par la fonction du signal d'entrée $U(s)$. Pour les courbes facilement reconnaissables, comme celle de la réponse indicielle d'un premier ordre, il est facile de déterminer le modèle.

b- Système naturellement stable

- Système à dominante du premier ordre avec retard :

Cette identification est basée sur la méthode mise en point par V. Broïda qui consiste à assimiler le système régulé à un système du premier ordre avec retard, dont la fonction de transfert est (s : variable de Laplace) :

$$H(s) = \frac{K e^{-Tt}}{\tau s + 1}$$

- K : gain statique du système
- T : temps mort (retard pure)
- τ : constante de temps

Pour déterminer ces coefficients, l'essai d'identification s'effectue en imposant un échelon de commande au système en boucle ouvert comme la montre la figure ci-dessous :

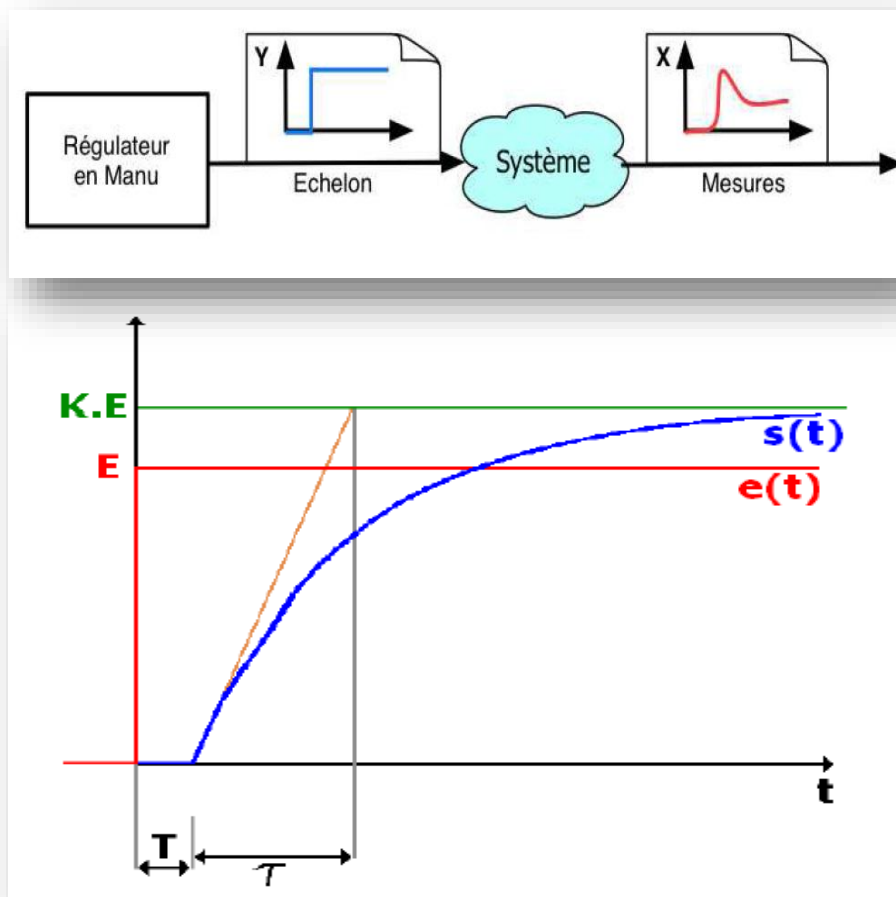


Figure III.10 : Schéma qui montre la méthode d'identification d'un premier ordre

c- Système naturellement instable

Quelle que soit la méthode employée, les paramètres du modèle du système à identifier sont ceux d'un intégrateur pur avec retard.

La fonction de transfert de ce modèle est la suivante :

$$H(s) = \frac{K e^{-Tt}}{s}$$

- K : gain statique du système
- T : temps mort (retard pure)

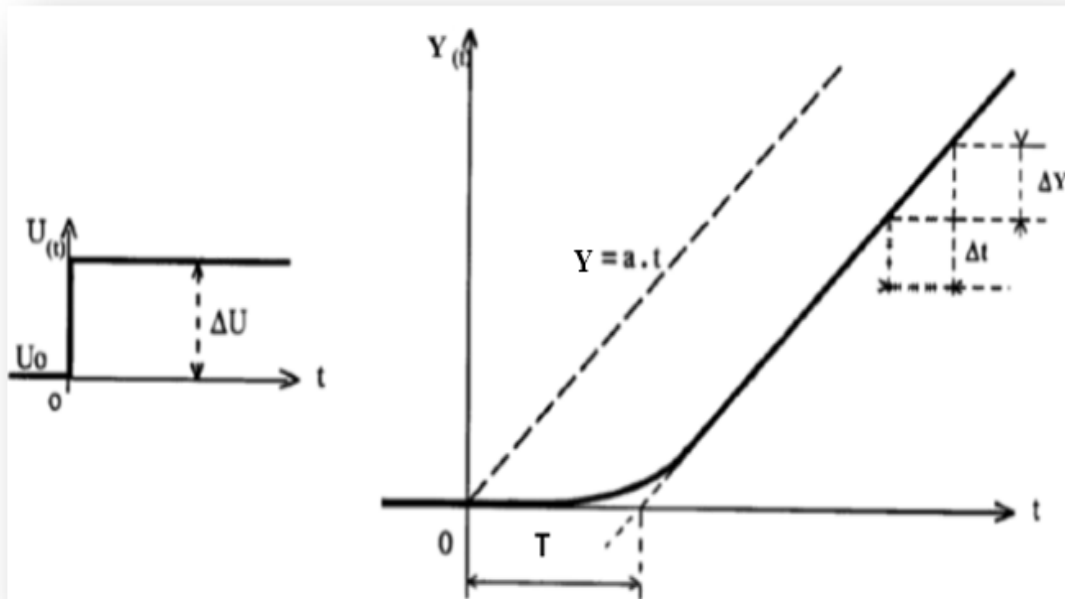


Figure III.11 : Schéma qui montre la méthode d'identification d'un intégrateur pur

- Le temps mort du modèle est déterminé graphiquement

- Coefficient d'intégration du procédé : $K = \frac{a}{\Delta U}$ (III. 1)

PARTIE C : Identification des différents systèmes (réglante-réglages) du CEE 117 process trainer

III.C.1. Le système de régulation de débit

En générale, la régulation de débit se fait par l'intermédiaire de deux types d'actionneurs considérés comme contrôleur :

- La pompe.
- La vanne proportionnelle.

III.C.1.1. Description du système de régulation de débit

Pour réguler le débit par les deux actionneurs cités précédemment, on va utiliser la partie droite de notre système qui comporte les éléments nécessaire, c'est-à-dire le circuit de fluide procédé.

La figure suivante montre cette partie en nommant ces différents composants :

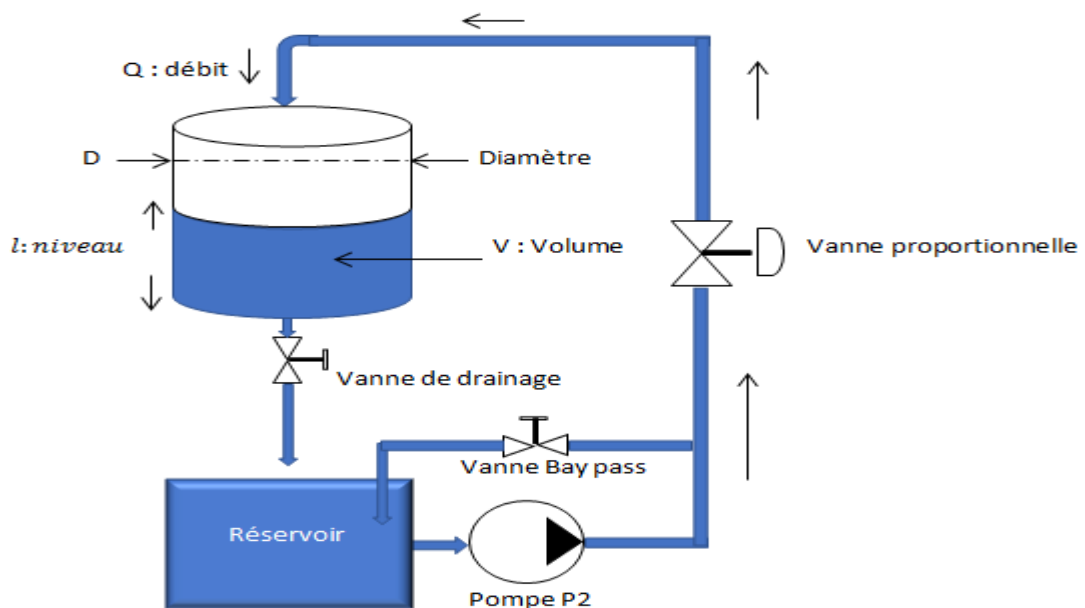


Figure III.12 : Schéma qui présente le circuit fluide procédé

Dans ce circuit, le fluide se dirige depuis le réservoir vers la cuve agitée par le biais de la pompe P2, et cela en traversant le refroidisseur, ensuite le fluide revient au réservoir par la gravité lorsque la vanne de drainage est ouverte. Notre système comporte deux capteurs de température avant et après le refroidisseur et un troisième capteur de débit, ainsi qu'une vanne By-pass qui représente la perturbation de notre système.

Dans ce qui va suivre on va étudier le contrôle continue du débit par les deux actionneurs.

En premier cas, la vanne proportionnelle reste entièrement ouverte et on régule le débit par le moyen de la pompe, cette dernière utilisée en mode analogique assure la variation de débit.

En deuxième cas, la pompe est utilisée à vitesse constante et on régule le débit par la vanne proportionnelle, dont la section de passage est infiniment ajustable, permet la variation de débit. Le débit de l'écoulement doit rester constant et ceci même en présence de perturbations. Dans notre système ces perturbations peuvent être l'effet de l'ouverture partielle ou complète de la vanne By-pass.

III.C.1.2. Identification du système pompe - débit

Pour l'identification de ce système, nous nous sommes basés sur l'étude de la réponse indicielle en boucle ouverte. On envoie un échelon à la pompe d'amplitude 80%, et à l'aide du logiciel CE2000, on relève la réponse qui est la tension image de débit selon la plage de conversion du capteur de débit.

En utilisant la méthode de broïda, on a assimilé la réponse de ce système à celui d'un premier ordre dont la fonction de transfert est donnée par :

$$H(s) = \frac{0.6072}{2.9612s+1} \quad (\text{III. 2})$$

Donc le modèle d'état est le suivant :

$$\begin{cases} \dot{x} = -0.3377x + 0.205u \\ y = x \end{cases} \quad (\text{III. 3})$$

La figure ci-dessus représente les résultats de l'identification :

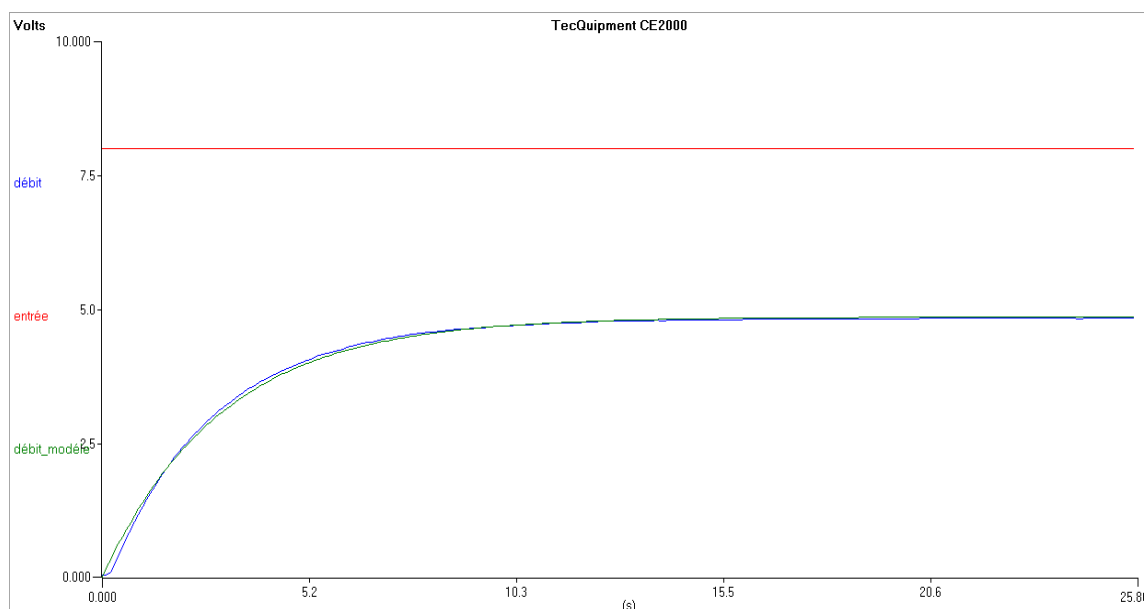


Figure III.13: Réponse réelle et identifiée du système pompe-débit en boucle ouverte

D'après la figure, les deux réponses, la réponse réelle et la réponse du modèle identifié, sont identiques, alors l'identification est validée.

III.C.1.3. Identification du système vanne – débit

Pour l'identification de ce système, nous nous sommes basés sur l'étude de la réponse indicielle en boucle ouverte. On envoie un échelon à la vanne proportionnel d'amplitude 100% et à l'aide du logiciel CE2000, on relève la réponse qui est la tension image de débit selon la plage de conversion du capteur de débit.

En utilisant la méthode de broïda, on a assimilé la réponse de ce système à celui d'un premier ordre dont la forme de la fonction de transfert est :

$$H(s) = \frac{0.6416}{2.7s+1} \quad (\text{III. 4})$$

Donc le modèle d'état est le suivant :

$$\begin{cases} \dot{x} = -0.3703x + 0.2376u \\ y = x \end{cases} \quad (\text{III. 5})$$

Pour valider le modèle identifié on a utilisé le logiciel CE2000, pour tracer la réponse réelle et du modèle identifié dans la même figure.

La figure ci-dessus représente les résultats de l'identification :

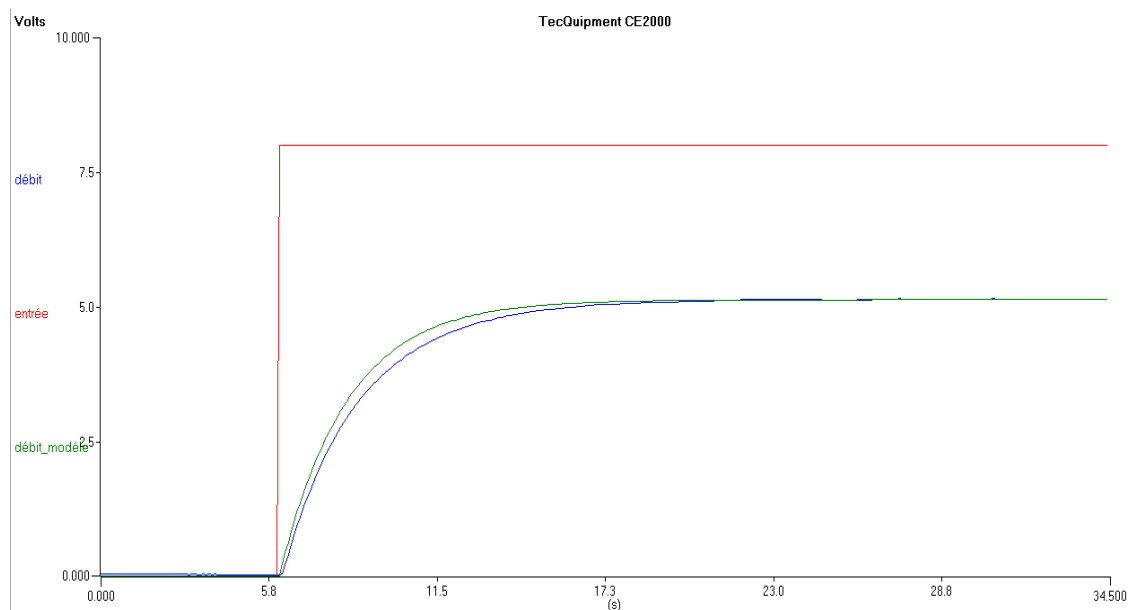


Figure III.14 : Réponse réelle et identifiée du système vanne-débit en boucle ouverte

D'après la figure, les deux réponses, la réponse réelle et la réponse du modèle identifié, sont presque identiques, alors l'identification est validée.

III.C.2. Le système de régulation de niveau :

En générale, la régulation de niveau se fait par deux types d'actionneurs. On peut utiliser la pompe comme actionneur comme on peut utiliser la vanne proportionnelle comme actionneur.

III.C.2.1. Description du système de régulation de niveau :

Pour régler le niveau de la cuve, nous allons utiliser que la partie droite de notre station CE117, c'est-à-dire le circuit de fluide procédé. Dans ce circuit la pompe P2 déplace de l'eau stockée dans le réservoir vers la cuve à travers un système de canalisation qui contient une vanne proportionnelle qui contrôle le débit comme le montre la figure suivante.

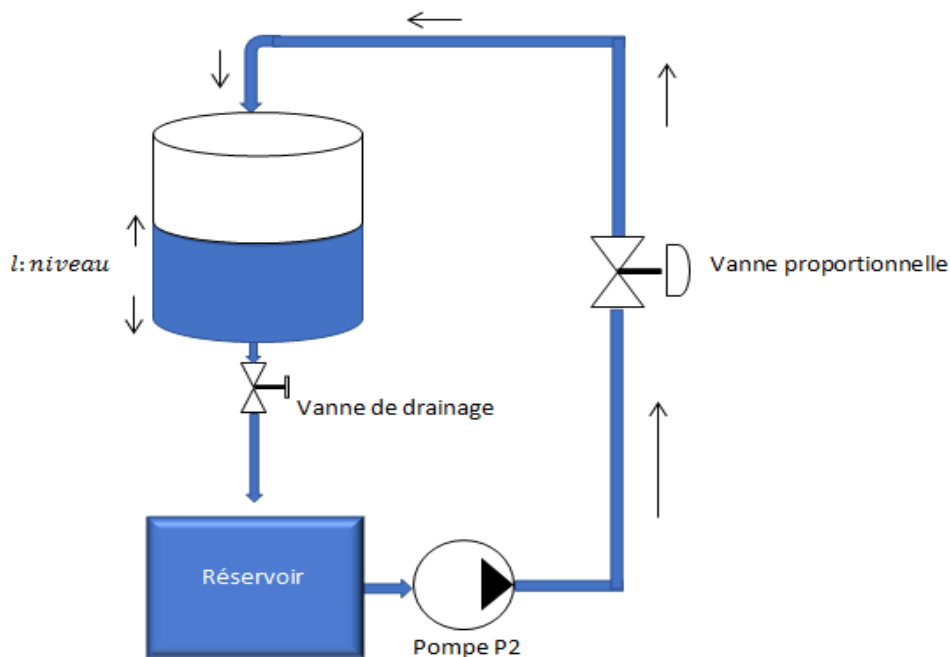


Figure III.15 : Schéma qui présente le circuit fluide procédé

Dans ce qui va suivre nous allons envisager deux types de régulation de niveau en utilisant les deux types d'actionneurs cités précédemment.

III.C.2.2. Identification du système pompe-niveau :

Dans cette partie on garde la vanne proportionnelle entièrement ouverte et on régule le niveau par le moyen de la pompe, cette dernière utilisé en mode analogique assure la variation de niveau de la cuve.

Pour l'identification de ce système, nous nous sommes basés sur l'étude de la réponse indicielle en boucle ouverte. On envoie un échelon à la pompe d'amplitude 80% et à l'aide du logiciel CE2000, on relève la réponse qui est la tension image du niveau.

En utilisant la méthode d'identification d'un système naturellement instable, on a assimilé la réponse de ce système à celui d'un intégrateur avec un retard pure dont la forme de la fonction de transfert est :

$$H(s) = \frac{Ke^{-Ts}}{s} \quad (\text{III. 6})$$

Après avoir calculé les différentes constantes du modèle, on donne la fonction de transfert du système pompe-niveau par l'expression:

$$H(s) = \frac{0.0626e^{-0.64s}}{s} \quad (\text{III. 7})$$

Pour l'obtention du modèle d'état du système on a fait deux approximations :

1. En négligeant le retard pur, on obtient le modèle suivant :

$$\begin{cases} \dot{x} = 0.0626u \\ y = x \end{cases} \quad (\text{III. 8})$$

2. En approximant le retard pur par un premier ordre, on obtient le modèle suivant :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -1.5625x_2 + 0.0978u \\ y = x_1 \end{cases} \quad (\text{III. 9})$$

Afin de valider le modèle trouvé on importe les résultats (réponse réel) ainsi le temps échantillonné sous forme un fichier « Bloc note » à partir de logicielle CE2000 et on simule la réponse des deux modèles réel et identifier sur la même figure.

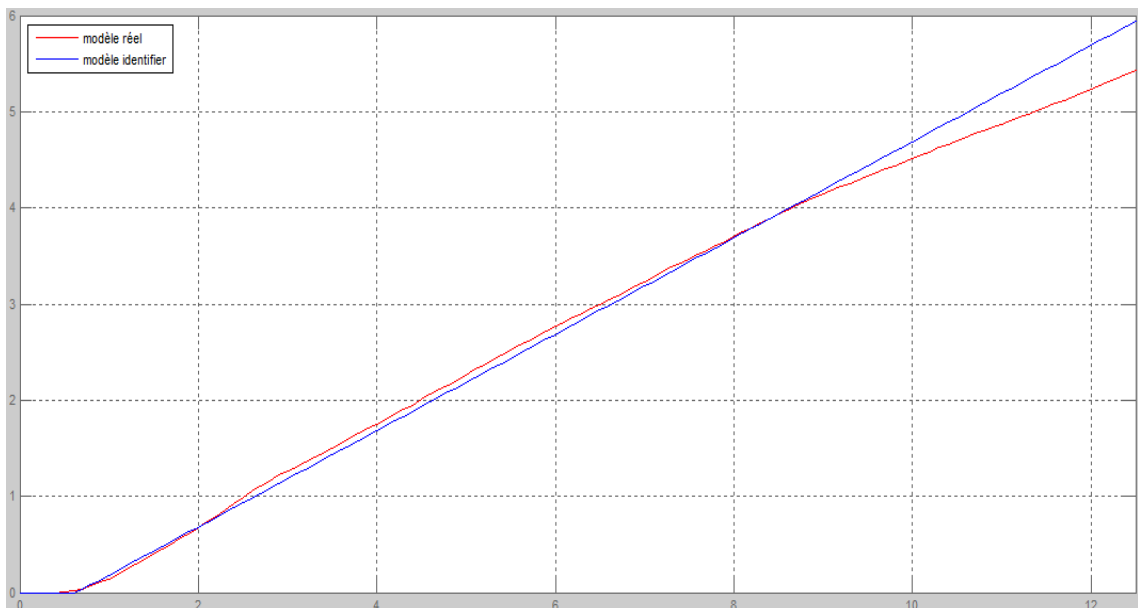


Figure III.16 : Réponse réelle et identifiée du système pompe-niveau en boucle ouverte

C.2.3. Identification du système vanne-niveau :

Cette fois ci on fixe la tension de commande de la pompe à 80%, et on régule le niveau par le taux d'ouverture de la vanne proportionnelle, cette dernière utilisé en mode analogique assure la variation de niveau de la cuve.

Pour l'identification de ce système, nous nous sommes basés sur l'étude de la réponse indicielle en boucle ouverte. On envoie un échelon à la pompe d'amplitude 80% et à l'aide du logiciel CE2000, on relève la réponse qui est la tension image de niveau.

En utilisant la méthode d'identification d'un système naturellement instable, on a assimilé la réponse de ce système à celui d'un intégrateur avec un retard pure qui a la forme suivante :

$$H(s) = \frac{Ke^{-Ts}}{s} \quad (\text{III.10})$$

Après avoir calculé les différentes constantes du modèle, la fonction de transfert du système pompe-niveau est donnée par l'expression suivante:

$$H(s) = \frac{0.0506e^{-0.61s}}{s} \quad (\text{III.11})$$

Pour l'obtention du modèle d'état du système on a fait deux approximations :

1. En négligeant le retard pur, on obtient le modèle suivant :

$$\begin{cases} \dot{x} = 0.0506u \\ y = x \end{cases} \quad (\text{III. 12})$$

2. En approximant le retard pur par un premier ordre, on obtient le modèle suivant :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -1.6393x_2 + 0.0829u \\ y = x_1 \end{cases} \quad (\text{III. 13})$$

Afin de valider le modèle trouvé on importe les résultats (réponse réel) ainsi le temps échantillonné sous forme un fichier « Bloc note » à partir de logicielle CE2000 et on simule la réponse des deux modèles réel et identifier sur la même figure.

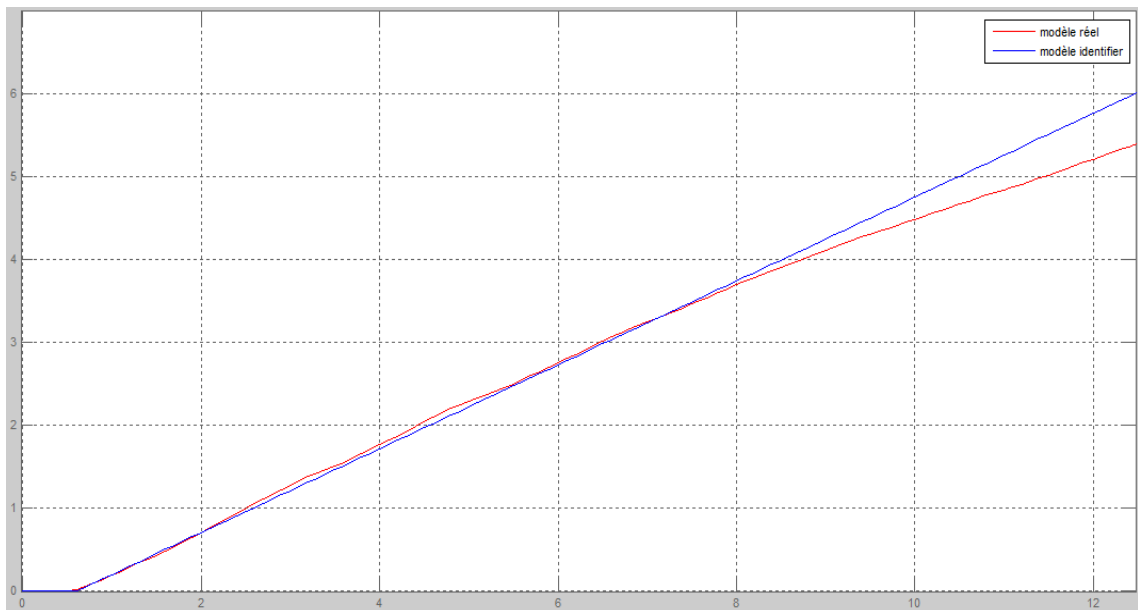


Figure III.17 : Réponse réelle et identifiée du système vanne-niveau en boucle ouverte

III.C.3. Le système de régulation de pression :

En générale, la régulation de pression se fait par deux types d'actionneurs. On peut utiliser la pompe comme actionneur comme on peut utiliser la vanne proportionnelle comme actionneur.

III.C.3.1. Description du système de régulation de pression :

La pression dans la cuve est une image du niveau de l'eau : quand le niveau augmente la pression augmente. Donc pour régler la pression dans la cuve, on aura besoins que de la partie droite de la station CE117, c'est-à-dire le circuit de fluide procédé. Dans ce circuit la pompe P2 déplace de l'eau stockée dans le réservoir vers la cuve ce qui va compresser l'air qui se trouve à l'intérieure comme le montre la figure suivante.

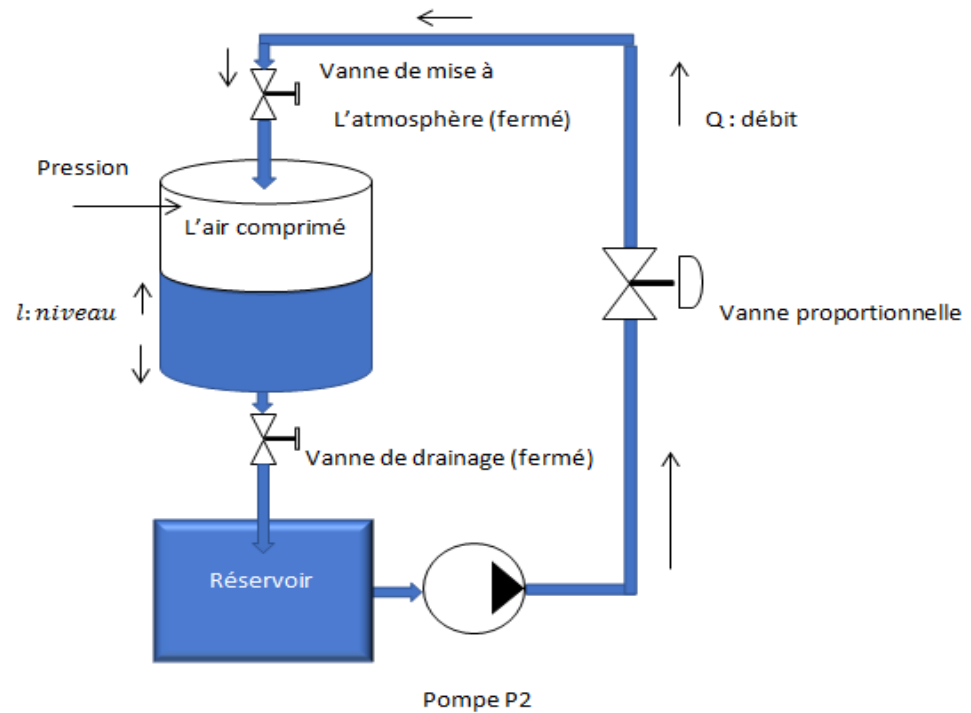


Figure III.18 : Schéma qui présente le circuit fluide procédé pour le réglage de la pression

Dans ce qui va suivre on va envisager deux types de régulation de pression en utilisant les deux types d'actionneur cité précédemment.

III.C.3.2. Identification du système pompe-pression:

Dans cette partie on garde la vanne proportionnelle entièrement ouverte et on remplit la cuve jusqu'à 5cm puis on ferme la vanne de la mise à l'atmosphère et on régule la pression par le débit de la pompe.

Pour l'identification de ce système, nous nous sommes basés sur l'étude de la réponse indicielle en boucle ouverte. On envoie un échelon à la pompe d'amplitude 8V et à l'aide du logiciel CE2000, on relève la réponse qui est la tension image de la pression.

En utilisant la méthode de broïda, on a assimilé la réponse de ce système à celui d'un premier ordre dont la forme de la fonction de transfert est :

$$H(s) = \frac{0.6125}{12s+1} \quad (\text{III. 14})$$

Le modèle d'état de ce système est donné par :

$$\begin{cases} \dot{x} = -0.0833x + 5.1041u \\ y = x \end{cases} \quad (\text{III. 15})$$

Afin de valider le modèle trouvé on importe les résultats (réponse réel) ainsi le temps échantillonné sous forme un fichier « Bloc note » à partir de logicielle CE2000 et on simule la réponse des deux modèles réel et identifier sur la même figure.

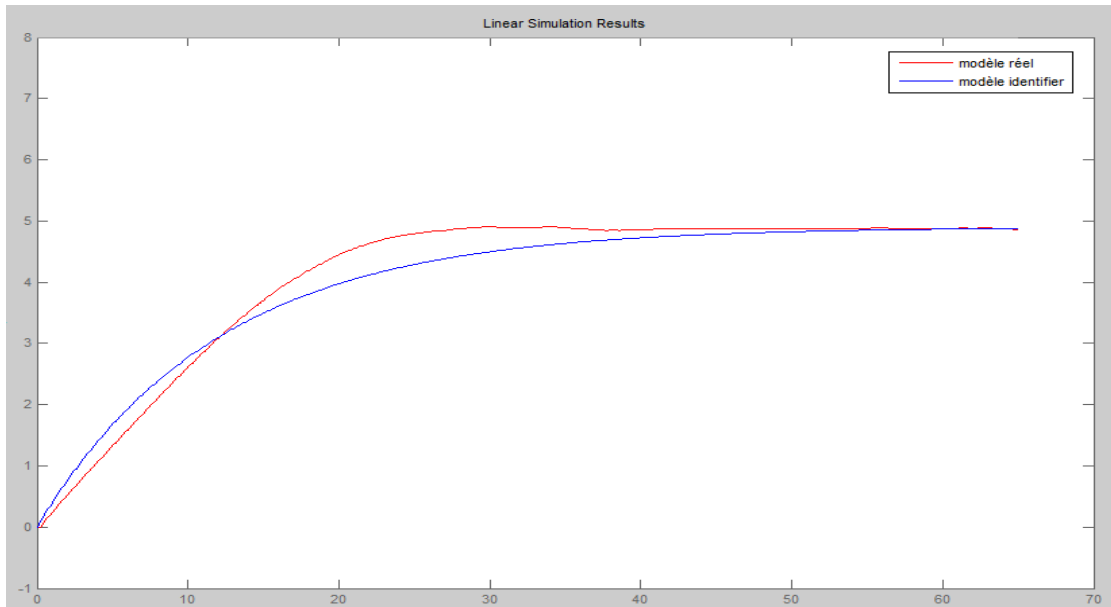


Figure III.19 : Réponse réelle et identifiée du système pompe-pression en boucle ouverte

C.3.3. Identification du système vanne-pression :

Cette fois ci on fixe la tension de commande de la pompe à 80% et puis on régule la pression par le taux d'ouverture de la vanne proportionnelle, cette dernière utilisé en mode analogique assurant la variation de pression de l'air dans la cuve.

Pour assurer la poursuite de la référence on va synthétiser un régulateur, mais pour le faire il faut d'abord avoir le modèle du système vanne-pression.

Pour l'identification de ce système, nous nous sommes basés sur l'étude de la réponse indicielle en boucle ouverte. On envoie un échelon de 10V à la vanne pour qu'elle s'ouvre à 100% et à l'aide du logiciel CE2000, on relève la réponse qui est la tension image de pression comme montre la figure ci-dessous.

D'après la figure ci-dessous on remarque bien que le système se comporte comme un intégrateur pur au départ, sauf que la réponse se stabilise c'est comme si le système est un premier ordre, mais ce n'est pas le cas puisque premièrement en réalité la pression se stabilise parce que le débit s'annule à cause de la pression de l'air comprimé qui a dépassé celle de débit. Et deuxièmement si on donne une tension nulle à la vanne la pression reste fixe ce qui n'est pas le cas d'un premier ordre.

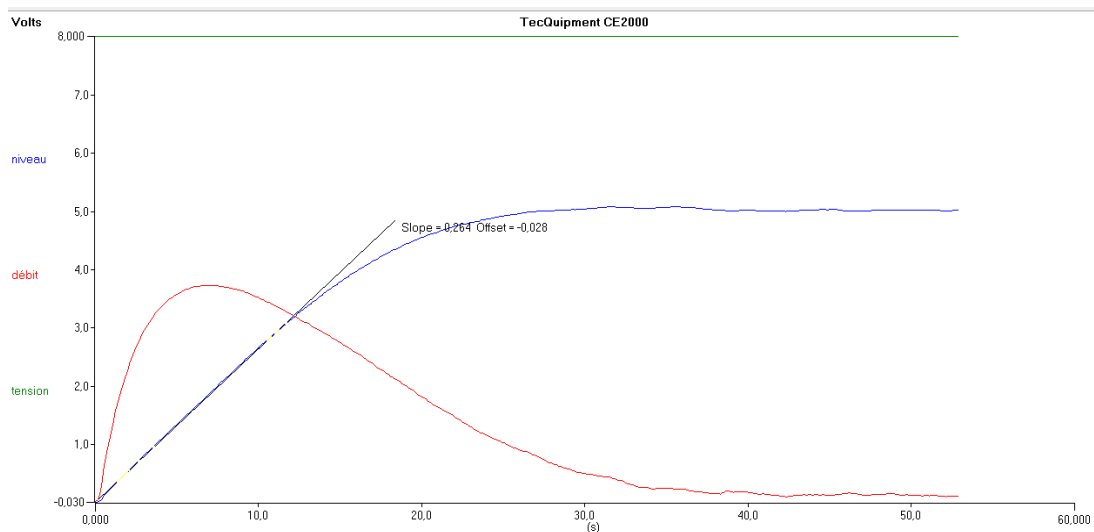


Figure III.20 : Réponse réelle du système vanne-pression en boucle ouverte

En utilisant la méthode d'identification d'un système naturellement instable cité précédemment, on a assimilé la réponse de ce système à celui d'un intégrateur avec un retard pure dont la forme de la fonction de transfert est :

$$H(s) = \frac{K}{s} \quad (\text{III. 16})$$

Après avoir calculé les différentes constantes du modèle on donne la fonction de transfert du système pompe-niveau par l'expression:

$$H(s) = \frac{0.0270}{s} \quad (\text{III. 37})$$

Le modèle d'état de ce système est donné par l'expression suivante :

$$\begin{cases} \dot{x} = 0.027u \\ y = x \end{cases} \quad (\text{III. 48})$$

Afin de valider le modèle trouvé on importe les résultats (réponse réel) ainsi le temps échantillonné sous forme un fichier « Bloc note » à partir de logicielle CE2000 et on simule la réponse des deux modèles réel et identifier sur la même figure.

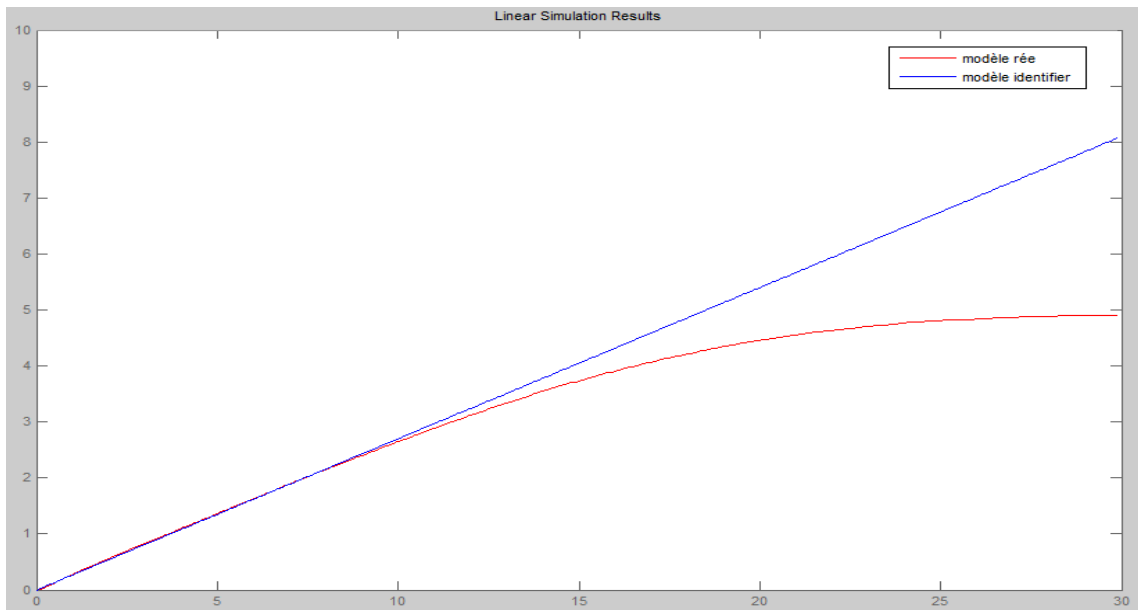


Figure III.21 : Réponse réelle et identifiée du système vanne-pression en boucle ouverte

III.C.4. Le système de régulation de température :

III.C.4.1. Description du système de régulation de température :

La température de réservoir de chauffe peut être réglé par une résistance d'échauffement et la pompe P1.

III.C.4.2. Identification du système pompe-température:

Comme le système est très lent alors il suffit d'appliqué une commande tout ou rien.

Conclusion :

Ce chapitre nous a permis d'avoir une bonne connaissance sur les différents systèmes qui existe sur la station CE117, ce qui nous aidera par la suite pour la régulation de chaque système.

CHAPITRE IV

**Régulation des différents systèmes
De la station Process Trainer (CE117).**

Introduction :

Ce chapitre sera partager en quatre parties, la première partie est une partie consacrer à la description des commandes qu'on va utiliser. Dans la deuxième partie on synthétisera des différentes commandes pour chaque système de la station, et dans la troisième partie on va présenter brièvement la méthode d'implémentation des commandes par le Step7 et la supervision par le WinCC, et on finira par une étude comparative entre les différentes commandes qui nous permettra de choisir les commandes les plus adéquates pour chaque système.

PARTIE A : Description des différentes commandes utilisées pour la synthèse des régulateurs

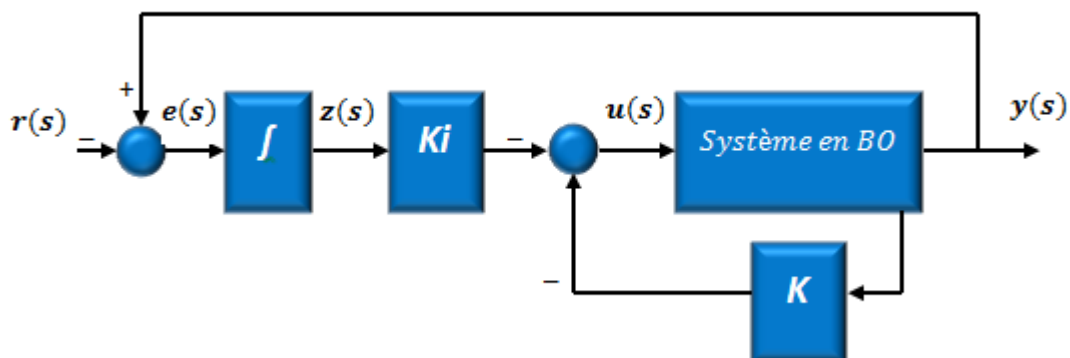
IV.A.1. Commande par retour d'état plus l'action intégrale :

Le retour d'état est utilisé pour modifier les pôles du système donc il nous permet de le stabiliser et de l'accélérer en boucle fermé. Le retour d'état n'assure pas seule la poursuite donc il faut lui rajouter une action intégrale pour avoir des bonnes performances.

Soit le système linéaire suivant :

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \quad (IV. 1)$$

Voici le schéma fonctionnel de la boucle fermé du système précédent commandé par le retour d'état plus l'action intégrale :



$$u = -kx - k_i \int_0^t e(\tau) d\tau \quad (\text{IV. 2})$$

$$\text{On pose : } z = \int_0^t e(\tau) d\tau \Rightarrow u = -kx - k_i z = -[k \ k_i] \Theta = -K\Theta. \quad (\text{IV. 3})$$

Avec $\Theta = \begin{pmatrix} x \\ z \end{pmatrix}$ l'état augmenté.

$$\begin{cases} \dot{\Theta} = \begin{pmatrix} \dot{x} \\ \dot{z} \end{pmatrix} = \begin{pmatrix} A & 0 \\ C & 0 \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u - \begin{pmatrix} 0 \\ 1 \end{pmatrix} r \\ y = (C \ 0) \begin{pmatrix} x \\ z \end{pmatrix} \end{cases} \Rightarrow \begin{cases} \dot{\Theta} = A_\Theta \Theta + B_\Theta u + B_r r \\ y = C_\Theta \Theta \end{cases} \quad (\text{IV. 4})$$

En remplaçant u par son expression $-K\Theta$ dans (IV. 5) on trouve :

$$\dot{\Theta} = (A_\Theta - B_\Theta K) \Theta + B_r r \quad (\text{IV. 6})$$

Pour modifier les valeurs propres de la matrice $(A_\Theta - B_\Theta K)$ et donc les pôles du système en boucle fermée, il suffit de modifier le gain K . Pour avoir le meilleur gain K , deux possibilités s'offre :

- Soit on utilise la fonction « *Place* » sur MATLAB sous la syntaxe suivante :
 $K = \text{place}(A_\Theta; B_\Theta; [\text{pole1}; \text{pole2}; \dots])$.
- Soit on calcule les valeurs propres de $(A_\Theta - B_\Theta K)$ et on les impose égale aux pôles voulu, pour obtenir des équations dont la solution est K qui convient.

IV.A.2. Commande par mode de glissement :

Pour éviter Lyapunov, on suppose qu'il existe une surface dans l'espace d'état décrite par l'équation $S(x_1, x_2, \dots, x_n) = 0$ sur laquelle les objectifs de commande sont satisfaits (stabilité et poursuite). Après avoir trouvé cette surface, le problème va devenir plus simple, il suffit de trouver la commande qui ramène le vecteur d'état sur cette surface et le maintenir.

A titre d'exemple, on prend le système suivant :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f(x_1, x_2) + g(x_1, x_2)u \end{cases} \quad (\text{IV. 7})$$

a. La première étape consiste à trouver la surface qui convient :

On prend l'ensemble de surface linéaire : $s_k = x_2 + kx_1$ (IV. 8)

$$\text{Pour } k=1 \Rightarrow s_1 = x_2 + x_1 \quad (\text{IV. 9})$$

$$\text{Pour } k=2 \Rightarrow s_2 = x_2 + 2x_1 \quad (\text{IV. 10})$$

$$\text{Pour } k=-1 \Rightarrow s_3 = x_2 - x_1 \quad (\text{IV. 11})$$

- Pour la surface S1 :

$$s_1 = 0 \Rightarrow x_2 = -x_1 \quad (\text{IV. 12})$$

En remplaçant (IV. 13) dans la première équation d'état on trouve :

$$\dot{x}_1 = -x_1 \Rightarrow x_1 \rightarrow 0 \Rightarrow x_2 \rightarrow 0 \text{ Alors } s_1=0 \text{ satisfaire les objectifs de la commande.}$$

- Pour la surface S2 :

$$s_2 = 0 \Rightarrow x_2 = -2x_1 \quad (\text{IV. 14})$$

En remplaçant (IV. 15) dans la première équation d'état on trouve :

$$\dot{x}_1 = -2x_1 \Rightarrow x_1 \rightarrow 0 \Rightarrow x_2 \rightarrow 0 \text{ Alors } s_2=0 \text{ satisfaire les objectifs de la commande.}$$

- Pour la surface S3 :

$$s_3 = 0 \Rightarrow x_2 = x_1 \quad (\text{IV. 16})$$

En remplaçant (IV. 17) dans la première équation d'état on trouve :

$$\dot{x}_1 = x_1 \Rightarrow x_1 \text{ diverge Alors } s_3=0 \text{ ne satisfait pas les objectifs de la commande.}$$

D'après ces exemples de surfaces on déduit qu'on peut trouver plus d'une surface qui satisfaire nos objectifs de commande.

- La surface de Slotine et Li (92) :

Pour le choix de la surface, Slotine et Li (92) ont proposées la forme suivante qui vérifie tout le temps la stabilité:

$$S = \left(\gamma + \frac{d}{dt}\right)^{r-1} e; \gamma > 0 \quad \begin{cases} e : \text{erreur qu'on veut stabilisé à } 0. \\ r : \text{est le degré relatif de } \frac{e}{u}. \end{cases} \quad (\text{IV. 18})$$

- La recherche de la commande :

Pour la synthèse de la commande il faut qu'elle assure les deux points suivant :

1. L'attractivité de la surface :

$$s \rightarrow 0 \Leftrightarrow \dot{s} s < 0 \quad (\text{IV. 19})$$

2. Le maintien (L'invariance):

$$\dot{s} = 0 \text{ pour } s = 0 \quad (\text{IV. 20})$$

On prend un exemple général :

Soit la représentation d'état d'un système quelconque :

$$\dot{x} = f(x) + g(x)u \quad (\text{IV. 21})$$

Et soit $S=0$ une surface satisfaisante.

Alors \dot{s} peut s'écrire sous la forme suivante :

$$\dot{s} = \frac{ds}{dx} [f(x) + g(x)u] \quad (\text{IV. 22})$$

Pour vérifier les deux points attractivité et invariance, on pose :

$$\dot{s} = -k \text{sign}(s) \Rightarrow \begin{cases} \dot{s} s < 0 \\ \dot{s} = 0 \text{ pour } s = 0 \end{cases} \quad (\text{IV. 23})$$

Et puis on tire la commande de cette équation :

$$u g = \left[\frac{ds(x)}{dx} g(x) \right]^{-1} \left[-\frac{ds}{dx} f(x) - k \text{sign}(s) \right] \quad (\text{IV. 24})$$

La fonction signe assure la robustesse de la commande, elle absorbe toutes les erreurs de modélisations, mais malheureusement elle engendre un problème majeur appelé phénomène de Chattering. Le phénomène de Chattering est dû à la discontinuité de la fonction signe. Pour remédier à ce problème il suffit de remplacer la fonction signe par l'une des deux fonctions suivantes : arc-tangente ou la tangente hyperbolique.

IV.A.3. La commande par logique floue :

Les méthodes conventionnelles de réglage sont basées sur une modélisation adéquate du système à régler et un traitement analytique à l'aide de la fonction de transfert ou d'équations d'état. Malheureusement, celles-ci ne sont pas toujours disponibles.

La majorité des systèmes industriels complexes sont difficiles à contrôler automatiquement.

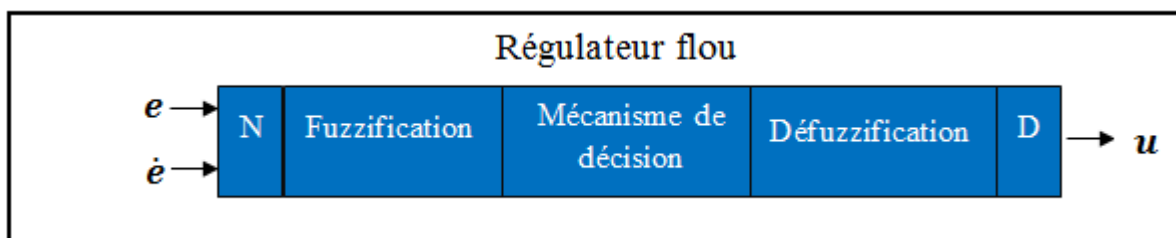
Cette difficulté provient de :

- leur non-linéarité,
- la variation de leurs paramètres,
- la qualité des variables mesurables.

Ces difficultés ont conduit à l'avènement et au développement de nouvelles techniques telles que la commande floue.

Régulateur floue :

On distingue classiquement trois parties dans la structure d'un contrôleur flou : la fuzzification, le mécanisme de décision et la défuzzification.



1. **La fuzzification** : est l'étape qui permet de transformer une grandeur mesurée sur le processus en des ensembles flous.

2. **Le mécanisme de décision** :

On distingue deux approches : l'approche de TSK et celle de Mamdani.

- Approche Mamdani : on calcule les commandes floues à partir des conditions floues suivant des règles données sous la forme :

Règle : Si condition floue Alors conclusion floue.

Pour cette approche, on est obligé de faire la Défuzzification de la conclusion pour avoir la commande réel.

- Approche TSK: on calcule la commande réelle à partir des conditions floues suivant des règles données sous la forme :

Règle : Si condition floue Alors conclusion réelle.

3.La défuzzification : Cette étape permet de transformer la commande floue, obtenu par le mécanisme de décision en une commande réel qui va être appliqué au processus. Cette étape est spécifique à approche de Mamdani.

4 .Les opérations de normalisation (N) et de dénormalisation (D) sont des étapes optionnelles.

IV.A.4. La commande adaptative indirecte:

Le modèle d'état des systèmes réel n'est jamais parfait, et ça à cause des variations paramétriques et des erreurs de modélisation.

Donc pour remédier à ce problème on a deux solutions possibles :

1. Commande robuste : on utilise une synthèse robuste p.ex. mode glissant, loop-shaping...etc.
2. La deuxième solution est la commande adaptative qui est une approche de la commande robuste.

Le modèle d'état est donné sous la forme suivante:

$$\begin{cases} \dot{x} = F(x, u, \theta) \\ y = h(x) \end{cases} \quad (\text{IV. 25})$$

θ : vecteur des paramètres il est inconnue

Les étapes de la commande adaptative :

1. La synthèse de la commande :

On suppose qu'on connaît le vecteur de paramètres du système et on synthétise la commande en utilisant l'une des méthodes de commande.

2. La synthèse de l'algorithme d'adaptation paramétrique (AAP) :

a- On choisit un signal y_i mesurable qui a la forme suivante :

$$y_i = \varphi^T \theta(t) \quad \text{Avec } \varphi : \text{vecteur de mesure.} \quad (\text{IV. 26})$$

b- On construit le prédicteur de y_i noté \hat{y}_i avec

$$\begin{cases} \hat{y}_i(t) = \varphi^T \hat{\theta}(t) \\ \hat{\theta}(t): \text{l'estimé à l'instant } t \text{ de } \theta(t) \end{cases} \quad (\text{IV. 27})$$

Alors l'erreur de prédiction sera:

$$\varepsilon = y_i - \hat{y}_i(t) = \varphi^T (\theta(t) - \hat{\theta}(t)) \quad (\text{IV. 28})$$

c- On choisit le critère J à optimiser :

$$J = \frac{1}{2} \varepsilon^2 \quad (\text{IV. 29})$$

Pour élaborer l'algorithme d'adaptation on utilise la descente de gradient :

$$\dot{\hat{\theta}} = -\gamma \left(\frac{dJ}{d\hat{\theta}} \right)^T \quad \text{Avec } \gamma > 0 \Rightarrow \dot{\hat{\theta}} = \gamma \cdot \varepsilon \cdot \varphi \quad (\text{IV. 30})$$

On fait une approximation du dérivé (dérivation numérique) :

$$\dot{\hat{\theta}} = \frac{\hat{\theta}(k+1) - \hat{\theta}(k)}{\Delta} \quad (\text{IV. 31})$$

Avec Δ : pas d'échantionnage.

D'après l'équation (IV. 32) et (IV. 33), on formule l'algorithme d'adaptation paramétrique suivant:

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \Delta \cdot \gamma \cdot \varepsilon \cdot \varphi \quad (\text{IV. 34})$$

3. La troisième étape consiste à remplacer le vecteur de paramètres par son estimateur $\hat{\theta}(k)$ dans l'expression de la commande synthétisé précédemment.
4. La quatrième et la dernière étape est l'implémentation :
 - a. On démarre l'algorithme toujours par les valeurs nominales.
 - b. Le Jell d'adaptation : il est utilisé dans le cas où un des paramètres risque de provoquer un problème de singularité, c'est-à-dire pendant les étirassions, si le paramètre arrive à une valeur de singularité, alors on l'affecte sa valeur précédente.

IV.A.5. Commande robuste par loop-shaping :

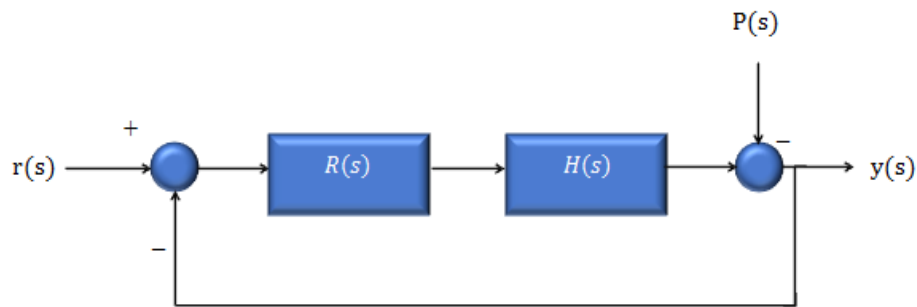
Dans une régulation on s'intéresse de plus à assurer la robustesse et les performances (rejet de perturbation et poursuite). En automatique il y a plusieurs méthodes pour faire une synthèse robuste parmi ces méthodes on trouve la synthèse par loop-shaping.

L'approche loop-shaping consiste en l'obtention d'une spécification relative à la boucle ouverte de l'asservissement à partir de spécifications relatives à divers transferts en boucle fermée. Parce qu'il est plus simple de travailler sur un unique transfert (la boucle ouverte) plutôt que sur une multitude de transferts bouclés, cette approche s'avère particulièrement adaptée au contexte industriel.

Robustesse :

Le système de commande est dit robuste vis-à-vis de l'erreur de modélisation ΔH , s'il arrive à assurer le cahier de charge non seulement sur le modèle nominale $H(s)$ mais aussi sur le système physique.

Soit un système de régulation en boucle fermé :



$$T_y(s) = \frac{H(s)R(s)}{1+H(s)R(s)}$$

$$S_y(s) = \frac{1}{1+H(s)R(s)}$$

$$G(s) = H(s)R(s)$$

$T_y(s)$: Fonction de transfert en boucle fermé (nominale).

$S_y(s)$: Fonction de sensibilité (nominale).

$G(s)$: Fonction de transfert en boucle ouverte(nominale).

$R(s)$: Fonction de transfert de régulateur.

ΔH : Erreur de modélisation de H .

D'après le théorème du « Small Gain » on tire la condition de stabilité (Robustesse) suivante :

$$\text{Gain}(\Delta H) \text{Gain}(T_y(s)) < 1 \quad \forall \omega \quad (\text{IV. 35})$$

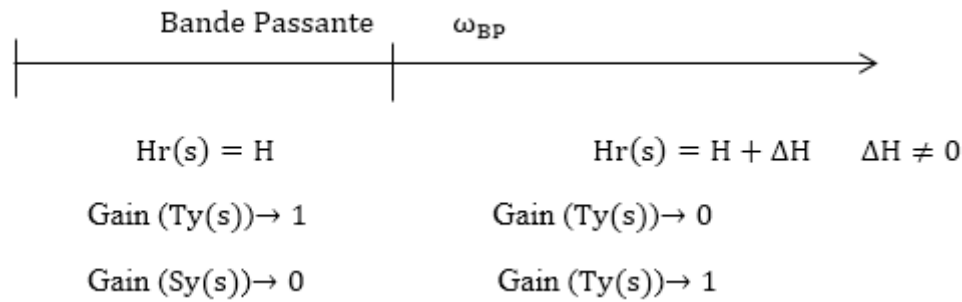
$$B(\omega) > \text{Gain}(\Delta H) \quad \forall \omega. \quad (\text{IV. 36})$$

Alors la condition (IV. 37) est vérifiée si

$$B(\omega) \cdot \text{Gain}(T_y(s)) < 1 \quad \forall \omega. \quad (\text{IV. 38})$$

Pour assurer que la condition de stabilité reste valable (la condition de robustesse) alors dans la plage de fréquence où $B(\omega) \gg 1$ il faut que $\text{Gain}(T_y(s)) \ll 1 \Rightarrow \text{Gain}(T_y(s)) \rightarrow 0$.

Mais comme $T_y(s) + S_y(s) = 1$ alors $S_y(s) \rightarrow 1$ ce qui n'est pas bien pour les performances. Mais en réalité ces erreurs ne sont considérables qu'à l'extérieur de la bande passante, là où on ne s'intéresse pas aux performances, donc à l'intérieure de la bande passante on peut assurer la poursuite ($\text{Gain}(T_y(s)) \rightarrow 1$) et le rejet de perturbation tout en gardant la robustesse.



3. La synthèse par loop-shaping :

Gabarit fréquentiel : Le gabarit fréquentiel transforme le cahier de charge en contraintes fréquentielles sur le gain de $G(s)$ (fonction de transfert du système avec le régulateur en boucle ouvert).

$$G(s) = H(s) \cdot R(s).$$

$R(s)$: le régulateur à déterminer.

A titre d'exemple, on prend le cahier de charge suivant et on fait la synthèse de régulateur par le loop-shaping :

1. Dans la bande passante erreur doit être inférieur à 1 %.
2. Erreur de modélisation est inférieur à +K dB au-delà de ω_{BP} .

Solution :

1. D'après le cahier de charge, on a:

$$e = r - y < 1\% r = 10^{-2}r \quad (\text{IV. 39})$$

$$\text{En négligeant } S_y(s) \text{ (} S_y(s) \approx 0 \text{), on trouve : } y = T_y r \quad (\text{IV. 40})$$

D'après (IV.32) et (IV.33) on trouve :

$$(1 - T_y)r < 10^{-2}r \Rightarrow |T_y| > (1 - 10^{-2}) = 0.99 \quad (\text{IV. 41})$$

Et comme $|T_y| = \left| \frac{G(s)}{1+G(s)} \right| > \frac{|G(s)|}{1+|G(s)|}$ Donc il suffit d'assurer :

$$\frac{|G(s)|}{1+|G(s)|} > 0.99 \quad (\text{IV. 42})$$

$$(\text{IV. 43}) \Rightarrow |G(s)| > 0.99 + 0.99|G(s)| \Rightarrow |G(s)| > 0.99 * 10^2.$$

Donc pour assurer une erreur $< 1\%$ dans la bande passante il suffit de prendre

$$|G(s)|_{db} > 40dB \quad (IV. 44)$$

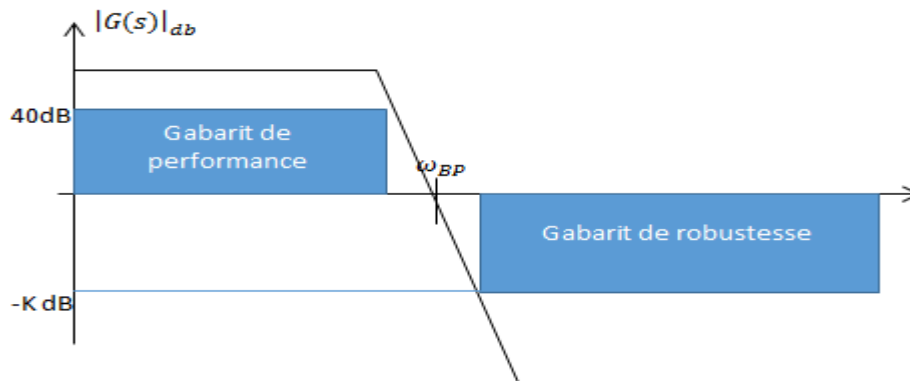
2. Pour assurer la robustesse il suffit d'assurer que :

$$|Ty(s)|_{db} + K < 0 \Rightarrow |Ty(s)|_{db} < -KdB \quad (IV. 45)$$

$$|Ty(s)| = \left| \frac{G(s)}{1+G(s)} \right| < |G(s)| \quad (IV. 46)$$

$$(IV.37) \text{ Et } (IV.38) \Rightarrow |G(s)|_{db} < -K \quad (IV. 47)$$

Donc pour assurer la robustesse, il suffit de prendre $|G(s)|_{db} < -KdB$ en dehors de la bande passante.



FigureIV.1: Diagramme de bode de la fonction de transfert en boucle ouvert qui montre les contraintes fréquentielles sur $G(s)$ sous formes des gabarits.

Finalement on choisit le régulateur qui assure les contraintes fréquentielles comme montre la figureIV.1 en utilisant le logiciel Matlab.

PARTIE B : Synthèse des régulateurs des différents systèmes

IV.B.1.Régulation de débit

a. Système pompe-débit

a.1.Régulation par PI

On synthétise le régulateur PI par la méthode de compensation, afin d'obtenir une réponse d'un premier ordre en boucle fermée :

Soit la fonction de transfert en boucle ouverte du système de régulation :

$$G(s) = R(s) * H(s) = \left[K_p \left(1 + \frac{1}{T_i s} \right) \right] \left[\frac{K}{\tau s + 1} \right] = \left[K_p \left(1 + \frac{1}{T_i s} \right) \right] \left[\frac{0.6072}{2.9612s + 1} \right] \quad (\text{IV. 48})$$

Afin de compenser le pôle du système, on prend $T_i = \tau$:

$$G(s) = \left[K_p \left(1 + \frac{1}{T_i s} \right) \right] \left[\frac{0.6072}{2.9612s + 1} \right] = \frac{K_p K}{\tau s} = \frac{a}{s}$$

La fonction de transfert en boucle fermée :

$$G_{bf}(s) = \frac{1}{\frac{1}{a}s + 1} = \frac{1}{T_0 s + 1} \quad (\text{IV. 49})$$

On prend $T_0 = \frac{\tau}{1,25}$ pour accélérer un petit peu le système en boucle fermée

Donc le régulateur PI vaut :

$$R(s) = 2.0586 \left(1 + \frac{1}{2.9612 s} \right)$$

(IV. 50)

a.1.1 Vérification de la robustesse par la méthode de loop-shaping :

Soit la fonction de transfert en boucle ouverte du système de régulation :

$$G(s) = R(s) * H(s) = \left[K_p \left(1 + \frac{1}{T_i s} \right) \right] \left[\frac{K}{\tau s + 1} \right] = \left[K_p \left(1 + \frac{1}{T_i s} \right) \right] \left[\frac{0.6072}{2.9612s + 1} \right]$$

1. Les performances (la poursuite et le rejet des perturbations) :

Le régulateur PI assure des bonnes performances (poursuite et rejet de perturbation), grâce au pôle nul qu'il impose en boucle ouverte, ce qui rend le gain $|G(s)|$ très grand dans la bande passante (pour les petites fréquences).

$$|G(s)| \gg 1 \Rightarrow |G(s)| \approx |G(s)| + 1.$$

$$\Rightarrow Ty(s) = \frac{G(s)}{1+G(s)} \approx 1 \Rightarrow \text{Bonne poursuite de la référence.}$$

Et comme $Ty(s) + Sy(s) = 1 \Rightarrow Sy(s) \approx 0 \Rightarrow$ un bon rejet de perturbation dans la bande passante.

2. La robustesse :

En générale et lors de l'identification c'est plus facile de déterminer le gain statique du système avec précision que de déterminer la constante du temps donc on aura une erreur sur la constante du temps, donc les erreurs de modélisation vont être en dehors de la bande passante.

Pour vérifier la robustesse, il faut d'abord estimer les erreurs de modélisation en dehors de la bande passante. Pour faire ceci, on va estimer la valeur la plus grande et la plus petite que peut prendre la constante du temps du système, puis on va tracer le diagramme de bode pour trois systèmes, le premier c'est le système identifié et le deuxième c'est le système avec la constante du temps la plus grande et le troisième sera le système avec la constante du temps la plus petite.

Et puis on va estimer l'erreur maximale entre le système identifié et les deux autres modèles.

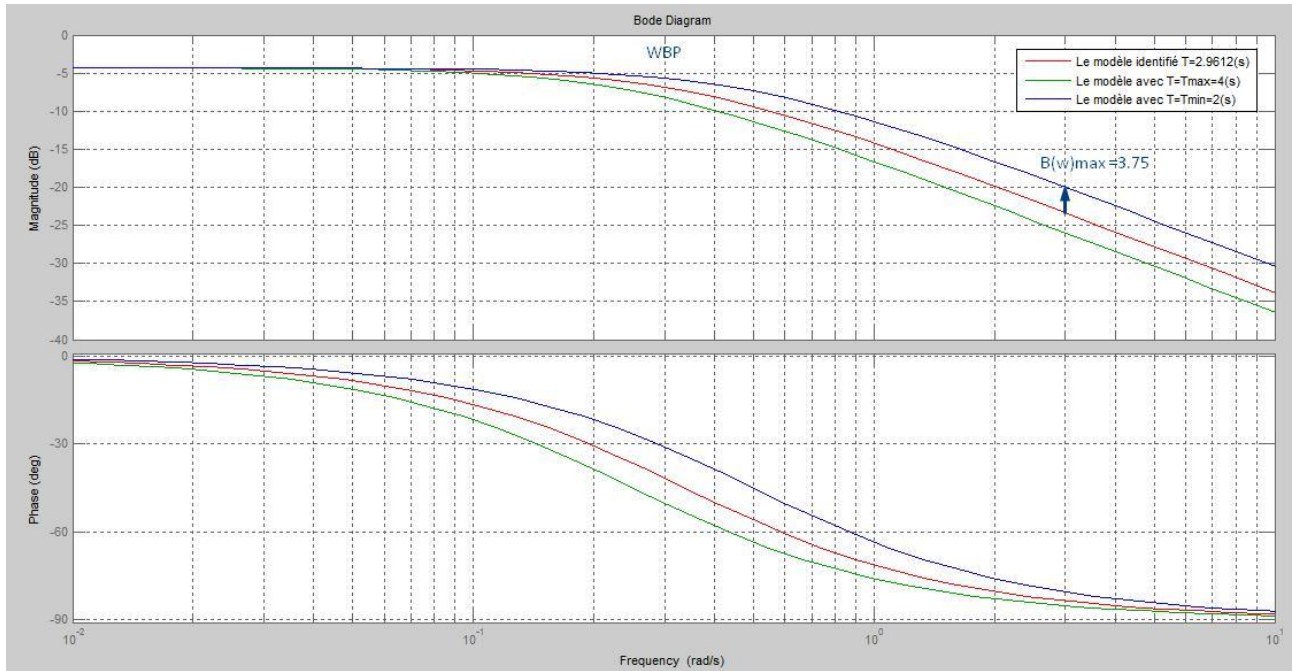


Figure IV.2: diagramme de bode qui montre les erreurs de modélisation

Les erreurs de modélisations en dehors de la bande passante sans inférieure à 3.75dB.

D’après le théorème du « Small Gain » :

Pour assurer la robustesse il faut assurer que $|Ty(s)|_{db} + 3.75 < 0$ pour les hautes fréquences (une décade après la fréquence de la bande passante).

$$|Ty(s)|_{db} < |G(s)|_{db} \tag{IV. 43}$$

Donc il suffit d’assurer :

$$|G(s)|_{db} + 3.75 < 0 \tag{IV.44}$$

$$\Rightarrow |G(s)|_{db} < -3.75 \text{ dB} \tag{IV.45}$$

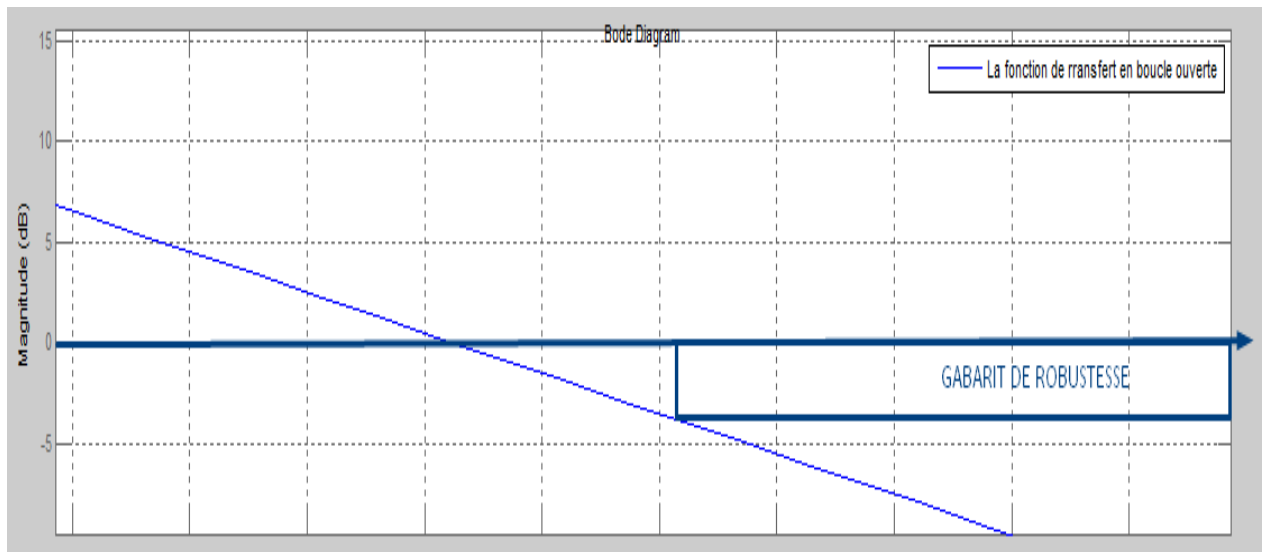


Figure IV.3 : diagramme de bode qui montre Le gabarit de robustesse

D’après le gabarit de robustesse on remarque bien que la relation (IV.45) est vérifiée pour $w \geq 0.65$ (rad/s).

La réponse du système réel sous le WinCC, montre que la commande est bien robuste avec une bonne poursuite de la référence et un bon rejet de perturbation.

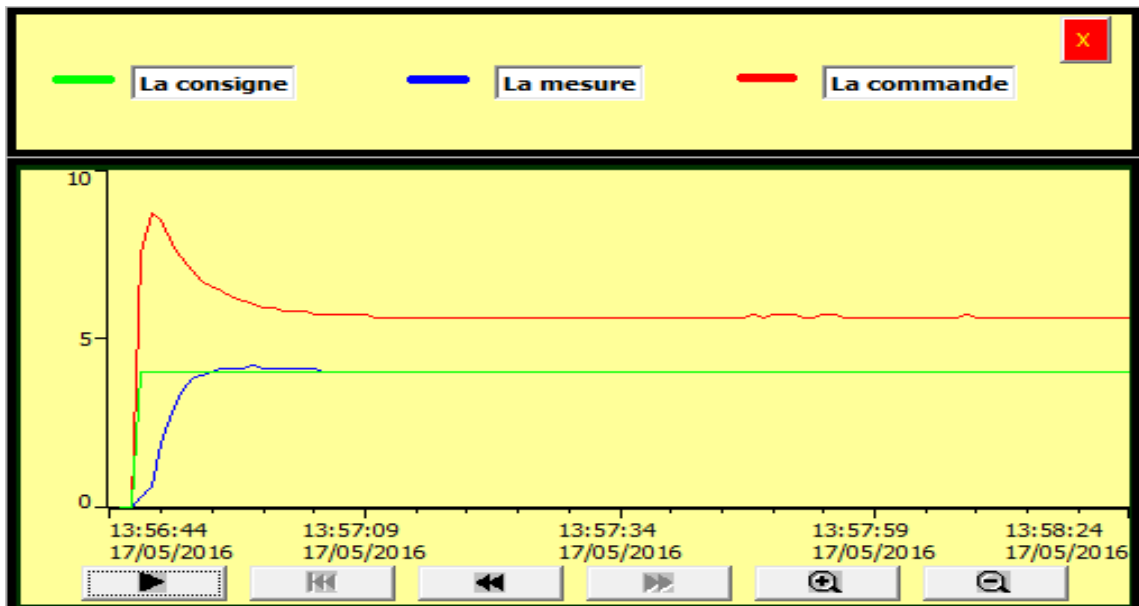


Figure IV.4: la réponse du système pompe-débit commandé par un PI sous WinCC

a.2.Régulation par mode de glissement

Pour synthétiser le régulateur par la méthode de mode de glissement, on utilise le modèle d'état qu'on a présenté dans le chapitre précédent :

$$\begin{cases} \dot{x} = -0.3377x + 0.2050u \\ y = x \end{cases} \quad (\text{IV. 51})$$

On choisit premièrement la surface sur laquelle les objectifs de commande sont satisfaits, pour cela on prend la surface de Slodine et Li : $S = (\gamma + \frac{d}{dt})^{r-1}e$; $\gamma > 0$

On prend $e = x - x_r$ et on obtient par la suite $S = e$

Ensuite, on cherche la commande pour laquelle on assure l'attractivité de la surface et le maintien ; il faut donc poser $\dot{s} = -K \text{sing}(s) \Rightarrow -0.3377x + 0.205u = -K \text{sing}(s)$ (IV. 52)

$$u = 1.6473x - 4.878 K \text{sign}(x - x_r) \quad (\text{IV. 53})$$

La figure suivante représente la réponse du système sous WinCC

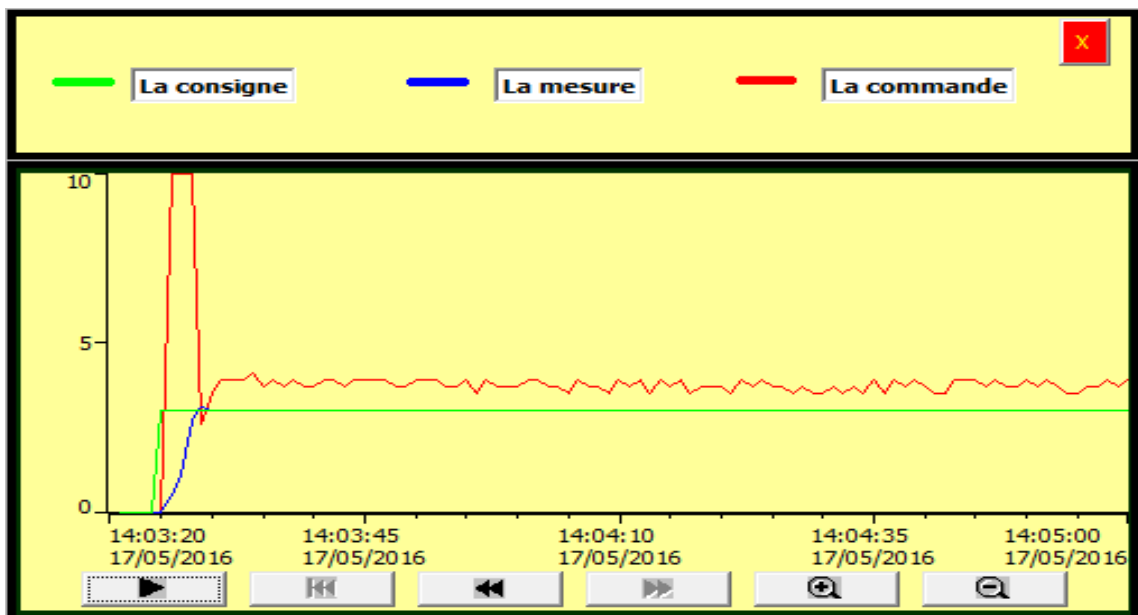


Figure IV.5 : La réponse du système pompe-débit commandé par mode glissant sous WinCC

a.3.Régulation par Retour d'état plus Action Intégrale

Le modèle d'état du système pompe-débit est donné par :

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \quad \text{Avec : } \begin{cases} A = -0.3377 \\ B = 0.2050 \\ C = 1 \end{cases} \quad (\text{IV. 54})$$

En appliquant un retour d'état plus action intégrale comme on a cité dans la description, on obtient le modèle suivant :

$$\dot{\theta} = (A_{\theta} - B_{\theta}K)\theta + B_r r \quad \text{Avec } \begin{cases} A_{\theta} = \begin{pmatrix} A & 0 \\ C & 0 \end{pmatrix} \\ B_{\theta} = \begin{pmatrix} B \\ 0 \end{pmatrix} \\ B_r = \begin{pmatrix} 0 \\ -1 \end{pmatrix} \end{cases} \quad (\text{IV. 55})$$

En imposant par le gain K les pôles en boucle fermé égale $\{-0.5 ; -0.35\}$, on obtient :

$$K = [k \quad ki] = [2.4990 \quad 0.8537]$$

(IV. 56)

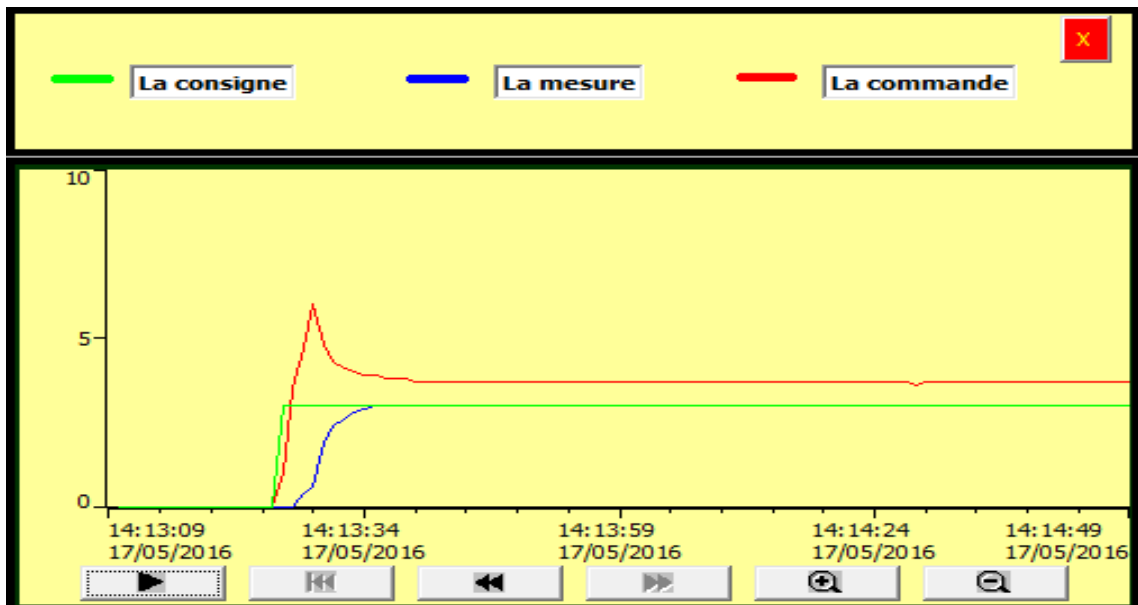


Figure IV.6: la réponse du système pompe-débit commandé par REI sous WINCC

b. Système vanne-débit

Pour ce système, on va synthétiser les commandes correspondantes par les mêmes méthodes appliquées pour le système pompe-débit.

b.1.Régulation par PI

Soit la fonction de transfert en boucle ouverte du système de régulation:

$$G(s) = R(s) * H(s) = \left[K_p \left(1 + \frac{1}{T_i s} \right) \right] \left[\frac{K}{\tau s + 1} \right] = \left[K_p \left(1 + \frac{1}{T_i s} \right) \right] \left[\frac{0.6416}{2.7s + 1} \right] \quad (\text{IV. 57})$$

De la même manière que le système pompe-débit, on synthétise le régulateur PI par la méthode de compensation :

$$\text{On prend } T_i = \tau \text{ et } T_0 = \frac{\tau}{1,25} \quad (\text{IV. 58})$$

On obtient le régulateur PI suivant :

$$R(s) = 1.9482 \left(1 + \frac{1}{2.7 s} \right) \quad (\text{IV. 59})$$

b.2.Régulation par mode de glissement

Pour synthétiser le régulateur, on utilise le modèle d'état:

$$\begin{cases} \dot{x} = -0.3703x + 0.2376u \\ y = x \end{cases} \quad (\text{IV. 60})$$

On obtient la commande suivante :

$$u = 1.5585 x - 4.2088 K \text{ sign}(x - x_r) \quad (\text{IV. 61})$$

b.3.Régulation par Retour d'état plus Action Intégrale

Le modèle d'état du système vanne-débit est donné par :

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \quad \text{Avec : } \begin{cases} A = -0.3703 \\ B = 0.2376 \\ C = 1 \end{cases} \quad (\text{IV. 62})$$

En appliquant les mêmes démarches qu'on a suivies pour le système pompe-débit, en imposant par le gain K les pôles en boucle fermé égale $\{-0.5 ; -0.9\}$, on obtient :

$$K = [k \quad k_i] = [4.3338 \quad 1.8939] \quad (\text{IV. 63})$$

IV.B.2.Régulation de niveau

- **Résolution de problème de non linéarité du capteur de niveau**

Comme on a cité dans le chapitre précédent, le capteur de niveau est non linéaire ce qui explique le fait que le niveau d'eau dans la cuve n'est pas en adéquation avec la consigne donnée pour la régulation. Pour résoudre ce problème, on va faire une adaptation par logique floue du gain de capteur, on suit les étapes suivant :

- 1- Fuzzification: On transforme la sortie du capteur y (Image de niveau en tension) en une variable floue y_F qui prend les valeurs floues suivantes : TP, P, MP, M, MG, G et TG avec chaque valeur est associé d'une fonction d'appartenance comme le montre la figure ci-dessous :

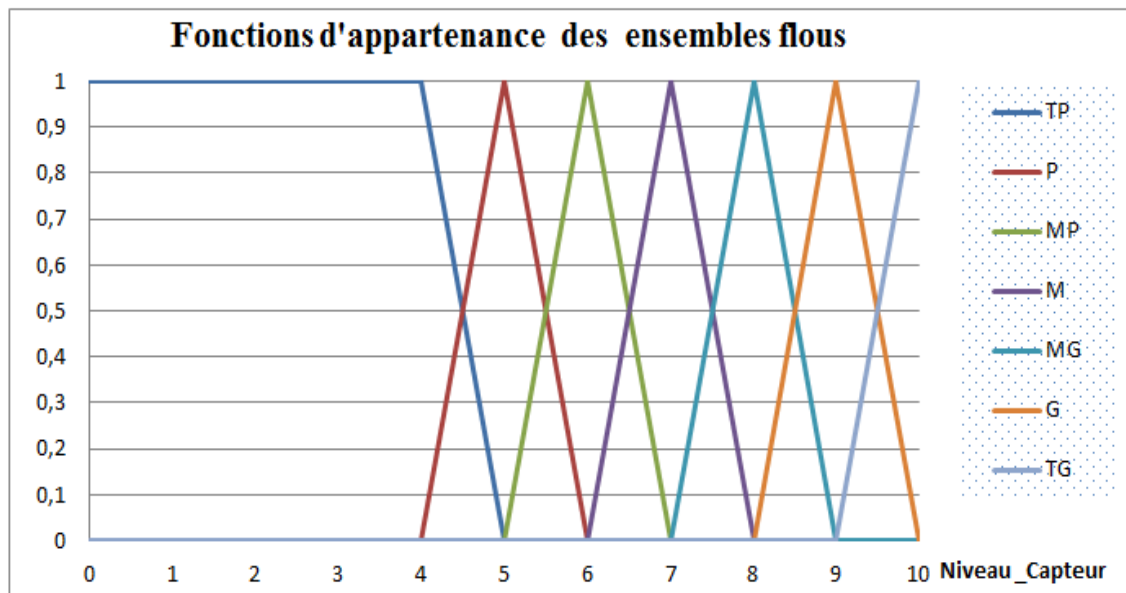


Figure IV.7: les fonctions d'appartenance de la variable floue y_F

- 2- On établit les règles sous l'approche de TSK :

Règle 1 : si y_F est KA alors $K_C = 1$

Règle 2 : si y_F est KB alors $K_C = 0.96$

Règle 3 : si y_F est KC alors $K_C = 0.9$

Règle 4 : si y_F est KD alors $K_C = 0.83$

Règle 5 : si y_F est KE alors $K_C = 0.8$

Règle 6 : si y_F est KF alors $K_C = 0.73$

Règle 7 : si y_F est KG alors $K_C = 0.67$

Soit α_i : le degré d'activation de la règle i.

$$\text{Règle 1} \rightarrow \alpha_1 = \mu_{TP}(y_F)$$

$$\text{Règle 2} \rightarrow \alpha_2 = \mu_P(y_F)$$

$$\text{Règle 3} \rightarrow \alpha_3 = \mu_{MP}(y_F)$$

$$\text{Règle 4} \rightarrow \alpha_4 = \mu_M(y_F)$$

$$\text{Règle 5} \rightarrow \alpha_5 = \mu_{MG}(y_F)$$

$$\text{Règle 6} \rightarrow \alpha_6 = \mu_G(y_F)$$

$$\text{Règle 7} \rightarrow \alpha_7 = \mu_{TG}(y_F)$$

D'après l'approche de TSK, Le gain réel du capteur est donné par l'expression suivante :

$$K = \frac{\sum \alpha_i K_i}{\sum \alpha_i} \quad (\text{IV. 64})$$

Donc pour avoir le niveau réel de la cuve, il suffit de diviser y par K

$$\text{Niveau_réel} = y(\text{Niveau_capteur})/K(\text{Gain_du_capteur})$$

Après la correction du gain du capteur le système se comporte exactement comme un intégrateur pur comme le montre la figure suivante :

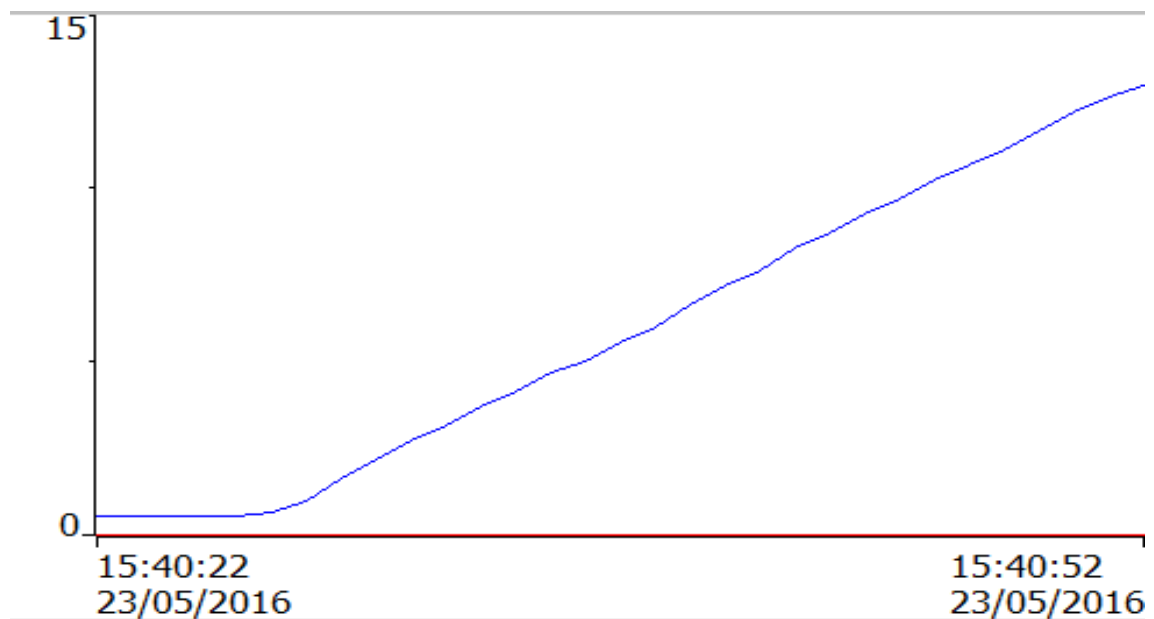
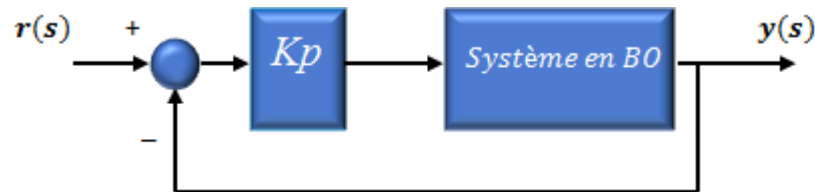


Figure IV.8: la réponse du système pompe-niveau après la correction du gain du capteur

a. Système pompe-niveau

a.1. Régulation par un gain proportionnel

Le système possède déjà un intégrateur, donc il suffit d'utiliser un régulateur proportionnel.



On approxime le retard pure par un système de première ordre :

$$H(s) = \frac{K}{s(1+Ts)} = \frac{0.0626}{s(1+0.64s)} \quad (\text{IV. 65})$$

La fonction de transfert en boucle fermé est donnée par la relation suivante :

$$G_{bf}(s) = \frac{H(s)}{1+H(s)} = \frac{KK_P}{Ts^2+s+KK_P} \quad (\text{IV. 66})$$

$$G_{bf}(s) = \frac{1}{\frac{1}{\omega_n^2}s^2 + \frac{2\xi}{\omega_n}s + 1} \quad (\text{IV. 67})$$

Par identification entre (IV.61) et (IV.62) on trouve :

$$\frac{1}{\omega_n^2} = \frac{T}{KK_P} \quad \frac{2\xi}{\omega_n} = \frac{1}{KK_P} \quad (\text{IV. 68})$$

D'après (IV.63) on trouve :

$$\omega_n = \frac{1}{2\xi T}; \quad K_P = \frac{\omega_n}{2\xi K} \quad (\text{IV. 69})$$

En imposant $\xi = 0.8$, on trouve :

$$\omega_n = 0.9765 \quad K_P = 9.7494 \quad (\text{IV. 70})$$

La fonction du transfert du régulateur est donnée par l'expression suivante:

$$R(s) = K_p = 9.7494$$

(IV. 71)

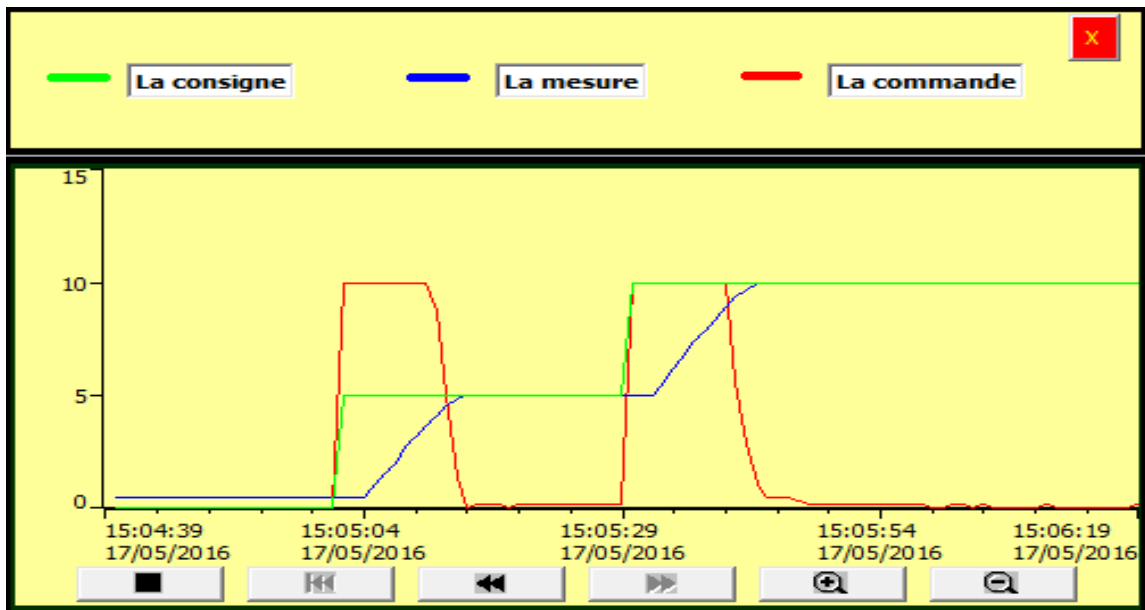


Figure IV.9: la réponse du système pompe-niveau commandé par un gain sous WINCC

a.2.Régulation par mode de glissement

- modèle d'état avec négligence du retard pur

Pour synthétiser le régulateur par la méthode du mode de glissement, on utilise le modèle d'état suivant dans lequel le retard est négligé :

$$\begin{cases} \dot{x} = 0.0626u \\ y = x \end{cases} \quad (\text{IV. 72})$$

On choisit premièrement la surface sur laquelle les objectifs de commande sont satisfaits, pour cela on prend la surface de Slotine et Li $S = (\gamma + \frac{d}{dt})^{r-1}e$; $\gamma > 0$.

On prend $e = x - x_r$ et on obtient par la suite $S = e$

Ensuite, on cherche la commande pour laquelle on assure l'attractivité de la surface et le maintien ; il faut donc poser $\dot{s} = -K \text{sing}(s) \Rightarrow 0.0626 u = -K \text{sing}(s)$ (IV. 73)

$$u = -15.9744 K \text{sign}(x - x_r)$$

(IV. 74)

La figure suivante représente la réponse obtenue après l'application de la commande calculée par mode de glissement.

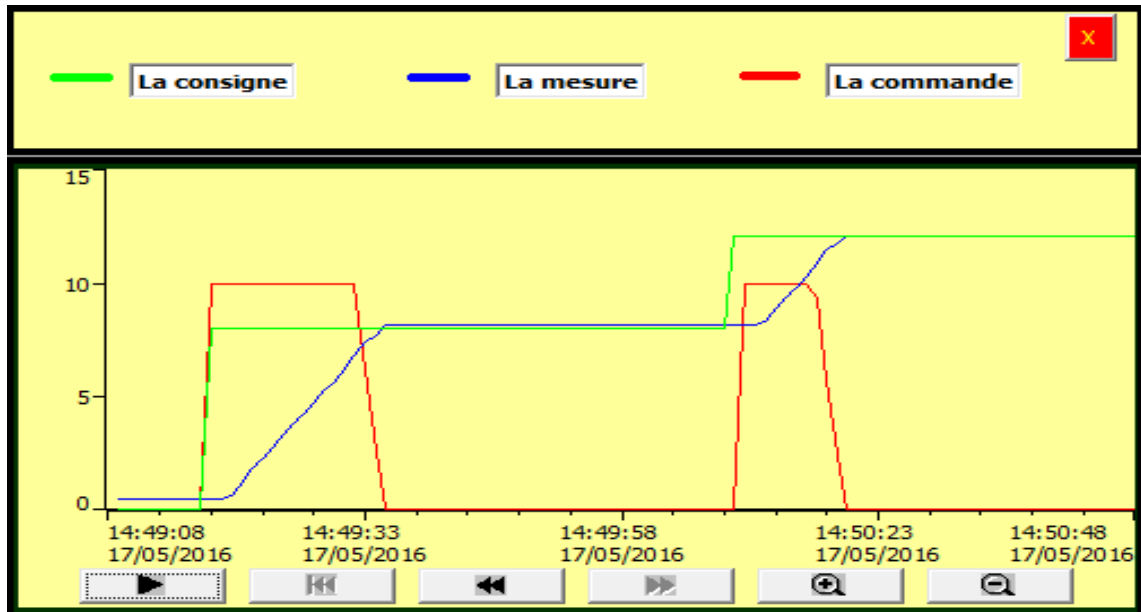


Figure IV.10: la réponse du système pompe-niveau commandé par mode glissant sous WINCC

- modèle d'état avec approximation du retard pur par un premier ordre :

Pour synthétiser le régulateur par la méthode de mode de glissement, on utilise le modèle d'état obtenu par l'approximation du retard pure du système par un premier ordre :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -1.5625x_2 + 0.0978u \\ y = x_1 \end{cases} \quad (IV. 75)$$

On choisit premièrement la surface sur laquelle les objectifs de commande sont satisfaits, pour cela on prend la surface de Slotine et Li : $S = (\gamma + \frac{d}{dt})^{r-1}e$; $\gamma > 0$

$$\text{Soit } e = x - x_r \Rightarrow r = 2 \Rightarrow S = \dot{e} + \gamma e = x_2 + \gamma(x_1 - x_{1r}) \quad (IV. 76)$$

Ensuite, on cherche la commande qui assure l'attractivité de la surface et le maintien ; il faut donc poser $\dot{s} = -K \text{sing}(s) \Rightarrow -1.5625x_2 + 0.0978u + \gamma x_2 = -K \text{sing}(s)$ (IV. 77)

$$u = \frac{1}{0.0978} [-(\gamma - 1.5625)x_2 - K \text{sign}(x_2 + \gamma(x_1 - x_{1r}))] \quad (IV. 78)$$

La réponse suivante est obtenue en prenant :

$$\begin{cases} K = 0.75 \\ \gamma = 1.5 \end{cases} \quad (IV. 79)$$

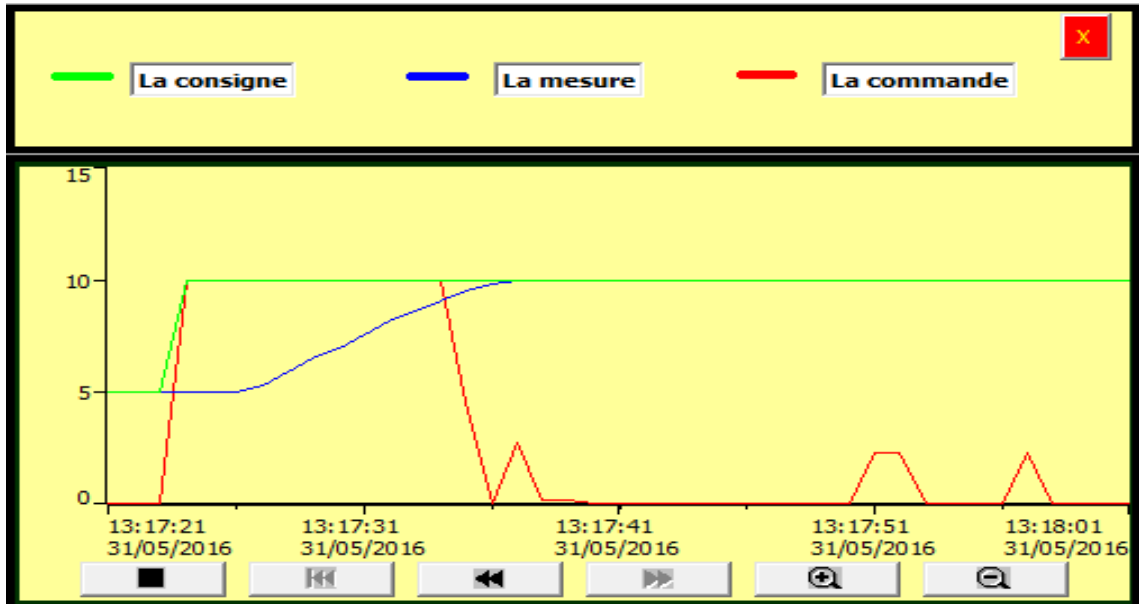
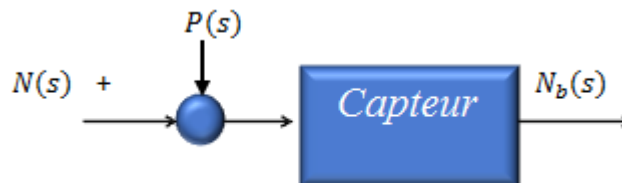


Figure IV.11: la réponse du système pompe-niveau commandé par mode glissant sous WINCC avant le filtrage

D’après la figure IV.11, on remarque bien qu’il y a une bonne poursuite, mais aussi la commande est bruitée.

4. Problème de bruit du capteur :

Le capteur de niveau n’est pas parfait, à cause de bruit qui perturbe la sortie du celui-ci. Lors de la dérivation de la sortie du capteur, l’effet du bruit devient plus important et peut provoquer des vrais problèmes lors de la régulation.



Le modèle d’état du système de réglage de niveau est donné sous la forme :

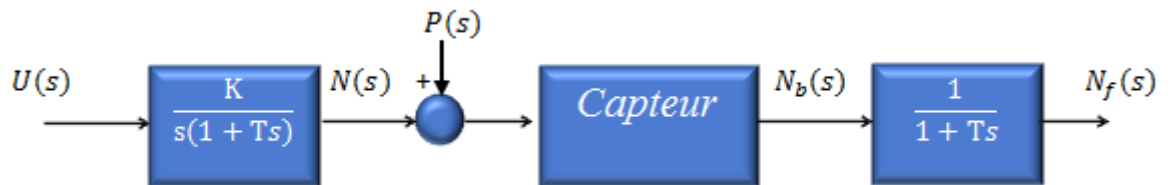
$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -ax_2 + bu \\ y = x_1 \end{cases} \quad (IV. 80)$$

La commande par mode de glissement de ce système est donné par :

$$u = \frac{1}{b} [-(\gamma - a)\dot{x}_1 - K \operatorname{sign}(\dot{x}_1 + \gamma(x_1 - x_{1r}))] \quad (\text{IV. 81})$$

Comme la commande dépend du dérivé de la sortie du capteur " x_1 ", donc elle dépend également du dérivé du bruit, ce dernier va perturber la commande en régime statique.

Donc pour éviter ce problème il suffit d'utiliser un filtre passe bas qui empêche le bruit de passer et de perturber la commande, et ça grâce au petit gain qu'il impose aux grandes fréquences.



Pour l'implémentation du filtre sur Step7, on propose les deux méthodes suivantes :

1^{ère} Méthode :

Pour cette méthode on va transformer la fonction de transfert en une équation différentielle qu'on va résoudre en utilisant l'intégration numérique :

$$x_f = \frac{x}{1+Ts} \Rightarrow x_f + T\dot{x}_f = x \quad (\text{IV. 82})$$

$$\Rightarrow \dot{x}_f = \frac{x - x_f}{T} = x_{df} \quad (\text{IV. 83})$$

$$\Rightarrow x_f = x_f + x_{df}T$$

$$(\text{IV. 84})$$

2^{ème} Méthode

Pour cette méthode on va approximer l'opérateur de dérivation par la forme suivante qui correspond à une dérivation numérique : $s = \frac{1-z^{-1}}{h}$ (IV. 85)

$$\begin{cases} x_f = \frac{x}{1+Ts} \\ s = \frac{1-z^{-1}}{h} \end{cases} \Rightarrow x_f = \frac{1}{1+T\left(\frac{1-z^{-1}}{h}\right)} x(z) \quad (\text{IV. 86})$$

$$= \frac{h}{h+T(1-z^{-1})} x(z) \quad (\text{IV. 87})$$

$$= \frac{\left(\frac{h}{T}\right)}{\left(\frac{h+T}{T}\right)z^{-1}} x(z) \quad (\text{IV. 88})$$

$$x_f(k) = \frac{T}{h+T} x_f(k-1) + \frac{h}{h+T} x(k) \quad (\text{IV. 89})$$

$$x_f(k) = \frac{T}{h+T} x_f(k-1) + \frac{h}{h+T} x(k) \quad (\text{IV. 90})$$

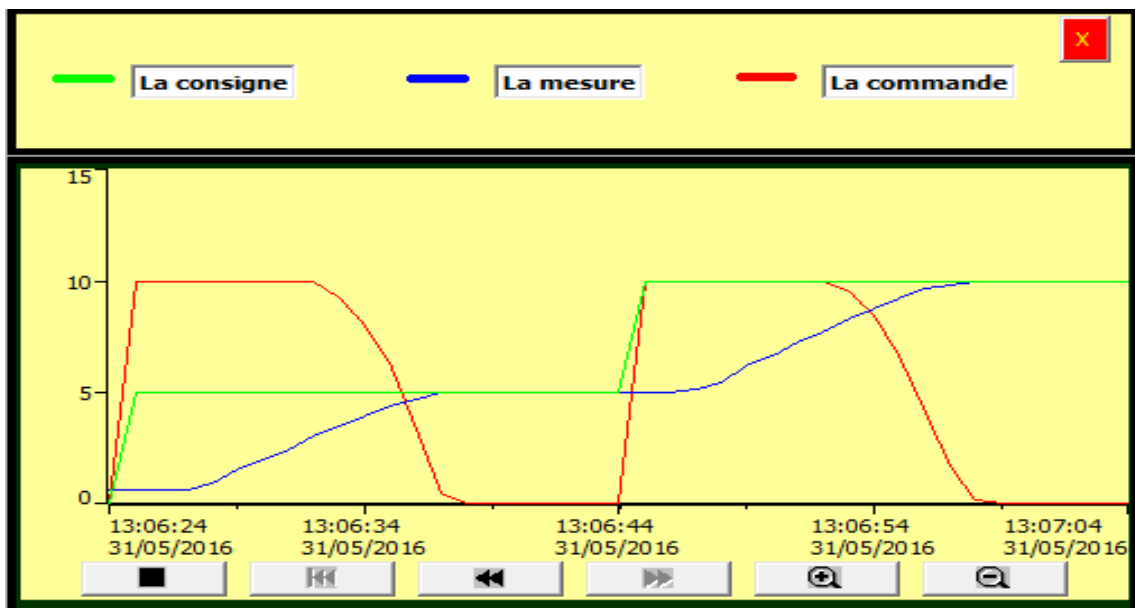


Figure IV.12: la réponse du système pompe-niveau commandé par mode glissant sous WINCC après le filtrage

a.3. Commande discrète :

Pour cette commande, on a élaboré un algorithme qui adapte la commande avec l'erreur par plage.

x_r : la consigne

x : la mesure

e : l'erreur de régulation

u : la commande

L'algorithme de réglage est le suivant :

$$e = x_r - x$$

Si $e \geq 3$ alors $u = 8 v$

Si non si ($e > 1$ et $e < 3$) alors $u = 6 v$

Si non si ($e > 0.5$ et $e \leq 1$) alors $u = 3 v$

Si non si ($e > 0$ et $e \leq 0.5$) alors $u = 2 v$

Si non $u = 0 v$

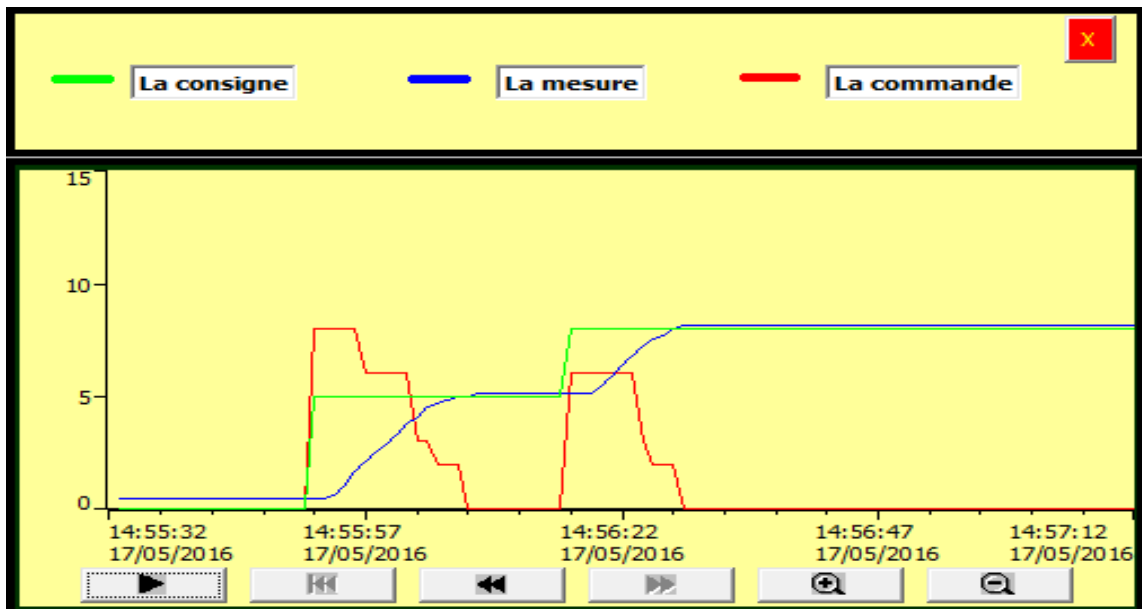
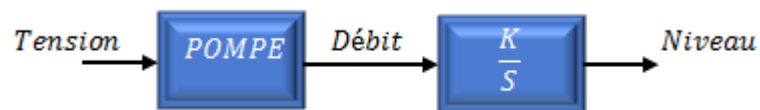


Figure IV.13: la réponse du système pompe-niveau commandé par la commande discrète sous WINCC

a.4. Régulation cascade :

La régulation cascade ressemble aux commandes synthétisées dans l'espace d'état parce qu'elle nous permet de maîtriser la dynamique de la boucle interne du procédé et de rajouter un degré de liberté en plus.

En réalité le système pompe-niveau n'est pas un intégrateur pur mais plutôt un premier ordre « système pompe-débit » en cascade avec un intégrateur pur qui représente le transfert entre le débit et le niveau comme montre la figure suivante :



Le problème qui se pose est de trouver le transfert Débit-Niveau :

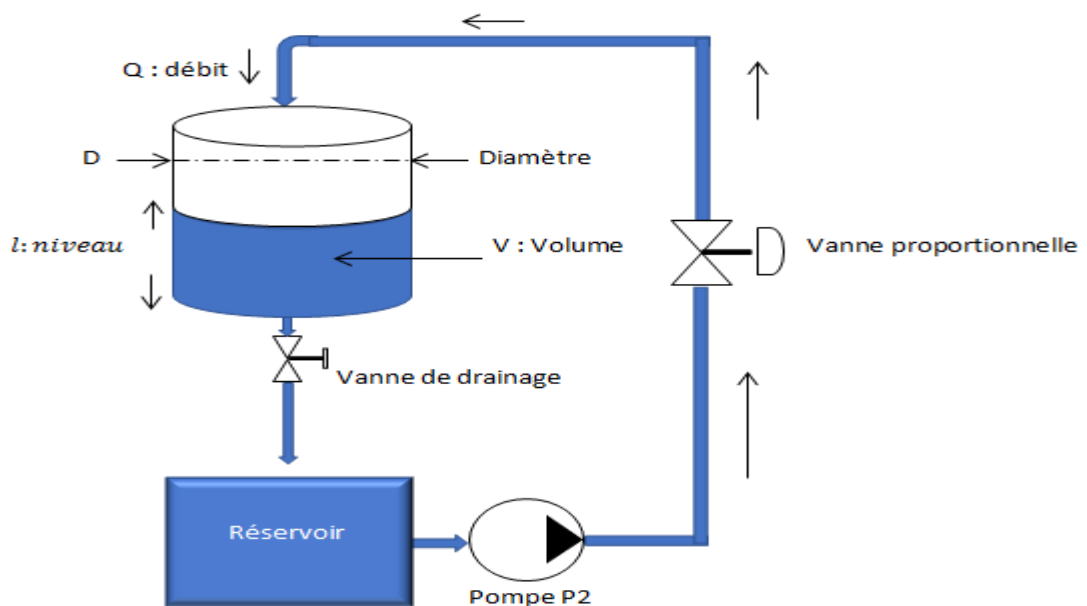


Figure IV.14 : schéma représentatif de circuit fluide procédés

Q: Débit d'eau qui rentre dans la cuve

l: Niveau d'eau dans la cuve

v: Volume d'eau dans la cuve

A: surface de la cuve

R: Rayon

Calcul de la surface de la cuve :

$$R = 8 \text{ cm} \Rightarrow A = \pi R^2 = 201.06 \text{ cm}^2 \quad (\text{IV. 91})$$

Calcul du transfert entre le débit et le niveau :

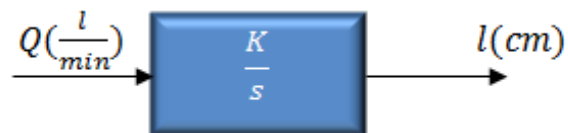
$$Q = \frac{dv}{dt} = A \frac{dl}{dt} \Rightarrow \frac{dl}{dt} = \frac{Q}{A} \quad (\text{IV. 92})$$

En appliquant la transformé de Laplace sur l'équation (IV. 93) on trouve :

$$sL(s) = \frac{Q(s)}{A} \Rightarrow \frac{L(s)}{Q(s)} = \frac{K}{s} \text{ avec } K = \frac{1}{A} \quad (\text{IV. 94})$$

D'après l'équation (IV. 95), on trouve la fonction de transfert entre le débit et le niveau :

$$H(s) = \frac{L(s)}{Q(s)} = \frac{K}{s} \quad (\text{IV. 96})$$



Donc pour avoir le niveau en « cm » il faut convertir le débit en $\left(\frac{\text{cm}^3}{\text{s}}\right)$, pour faire ça il suffit de multiplier le débit par un gain :

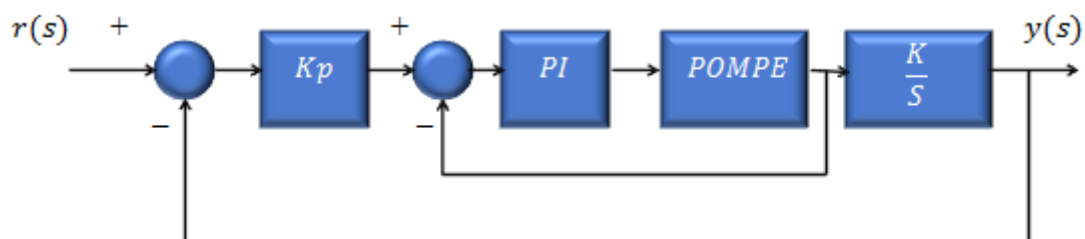
$$K1 = \frac{10^3}{60} \quad (\text{IV. 97})$$

$$(\text{IV. 98}) \text{ et } (\text{IV. 88}) \Rightarrow K' = KK1 = \frac{K1}{A} = 0.08 \quad (\text{IV. 99})$$

Donc le transfert entre le débit et le niveau est donnée par l'expression suivante :

$$H(s) = \frac{L(s)}{Q(s)} = \frac{0.08}{s} \quad (\text{IV. 100})$$

Une fois trouvé tous les fonctions de transferts, il nous reste qu'à calculer le régulateur de la boucle interne puis celui de la boucle principale selon la figure suivante :



La synthèse de PI de la boucle interne :

La boucle interne de ce système est la même que celle du système pompe-débit en boucle fermé, alors on garde le même régulateur PI qu'on a synthétisé.

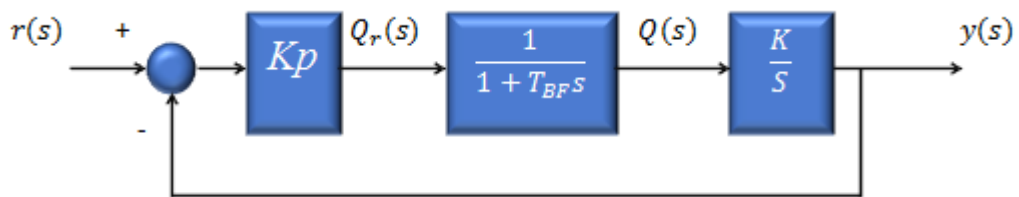
$$R(s) = 2.0586 \left(1 + \frac{1}{2.9612s} \right) \quad (\text{IV. 101})$$

La fonction du transfert de la boucle interne est donc donnée par l'expression suivante :

$$Ty(s) = \frac{1}{1+T_{BF}s} = \frac{1}{1+2.3689s} \quad (\text{IV. 102})$$

La synthèse du gain proportionnel de la boucle principale (Kp):

Puisque le système possède déjà un intégrateur. Alors il suffit d'utiliser une action proportionnelle.



La fonction du transfert en boucle ouverte est donnée par la relation suivante :

$$G(s) = \frac{K_P K}{s(1+T_{BF}s)} = \frac{0.08K_P}{s(1+2.3689s)} \quad (\text{IV. 103})$$

La fonction de transfert en boucle fermée est donnée par la relation suivante :

$$G_{bf}(s) = \frac{G(s)}{1+G(s)} = \frac{KK_P}{T_{BF}s^2+s+KK_P} \quad (\text{IV. 104})$$

$$G_{bf}(s) = \frac{1}{\frac{1}{\omega_n^2}s^2 + \frac{2\xi}{\omega_n}s + 1} \quad (\text{IV. 105})$$

Par identification entre (IV.95) et (IV.96) on trouve :

$$\frac{1}{\omega_n^2} = \frac{T_{BF}}{KK_P} \quad \frac{2\xi}{\omega_n} = \frac{1}{KK_P} \quad (\text{IV. 106})$$

D'après les deux équations trouvées précédemment on a :

$$\omega_n = \frac{1}{2\varepsilon T_{BF}} ; \quad K_P = \frac{\omega_n}{2\varepsilon K} \quad (\text{IV. 107})$$

En imposant $\varepsilon = 0.7$ pour avoir une meilleure réponse on trouve :

$$\omega_n = 0.3015 \quad K_P = 2.6919 \quad (\text{IV. 108})$$

La fonction du transfert du régulateur est donnée par l'expression suivante:

$$R(s) = K_P = 2.6919$$

(IV. 109)

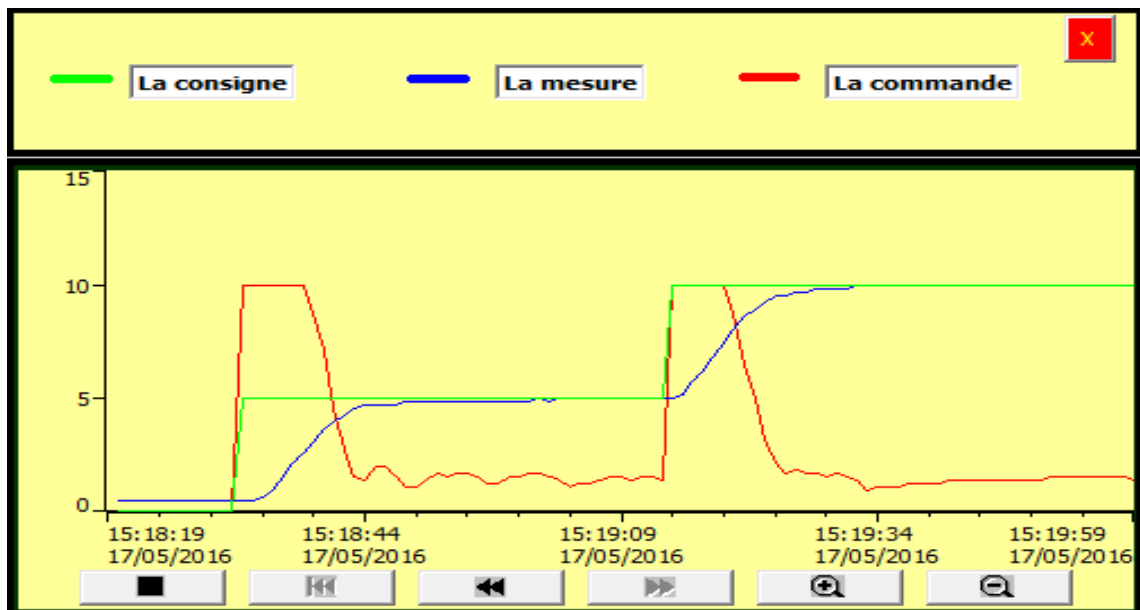


Figure IV.15: la réponse du système pompe-niveau commandé en cascade sous WINCC

b. système vanne-niveau

b.1. Régulation par un gain proportionnel

Le système possède déjà un intégrateur c'est pour cela on utilise que l'action proportionnelle.

On approxime le retard pure par un système de première ordre :

$$F(s) = \frac{K}{s(1+Ts)} \quad (\text{IV. 110})$$

On synthétise le régulateur proportionnel de la même manière que le système pompe-niveau, on obtient le résultat suivant :

$$R(s) = K_P = 12.6543$$

(IV. 111)

b.2. Régulation par mode de glissement

- modèle d'état avec négligence du retard pur

Pour synthétiser le régulateur par la méthode de mode de glissement, on utilise le modèle d'état

$$\text{suisant : } \begin{cases} \dot{x} = 0.0506u \\ y = x \end{cases} \quad (\text{IV. 112})$$

On obtient la commande suivante :

$$u = -19.7628 K \text{ sign}(x - x_r) \quad (\text{IV. 113})$$

- modèle d'état avec approximation du retard pur par un premier ordre

Pour synthétiser le régulateur par la méthode de mode de glissement, on utilise le modèle d'état obtenu par l'approximation du retard pure du système par un premier ordre :

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -1.6393x_2 + 0.0829u \\ y = x_1 \end{cases} \quad (\text{IV. 114})$$

On choisit premièrement la surface sur laquelle les objectifs de commande sont satisfaits, pour cela on prend la surface de Slodine et Li : $S = (\gamma + \frac{d}{dt})^{r-1}e$; $\gamma > 0$

$$e = x - x_r \Rightarrow r = 2 \Rightarrow S = \dot{e} + \gamma e = x_2 + \gamma(x_1 - x_{1r}) \quad (\text{IV. 115})$$

Ensuite, on cherche la commande qui assure l'attractivité de la surface et le maintien ; il faut donc poser $\dot{s} = -K \text{ sing}(s) \Rightarrow -1.5625x_2 + 0.0978u + \gamma x_2 = -K \text{ sing}(s)$ (IV. 116)

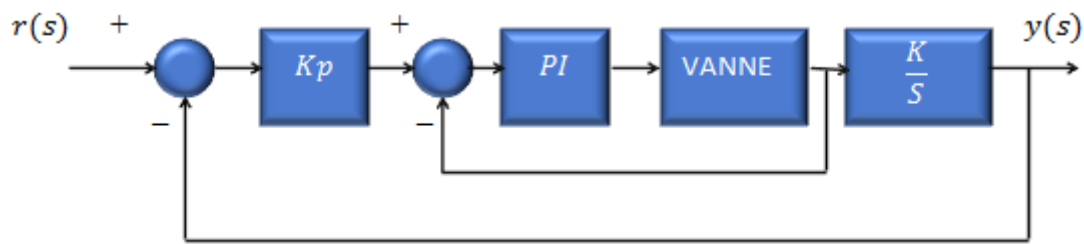
$$u = \frac{1}{0.0829} [-(\gamma - 1.6393)x_2 - K \text{ sign}(x_2 + \gamma(x_1 - x_{1r}))] \quad (\text{IV. 117})$$

b.3. Régulation cascade

D'après les développements qu'on a faits, on a trouvé que le transfert entre le débit et le niveau est donnée par l'expression suivante :

$$H(s) = \frac{L(s)}{Q(s)} = \frac{0.08}{s} \quad (\text{IV. 118})$$

Une fois trouvé tous les fonctions de transferts il nous reste qu'à calculer le régulateur de la boucle interne puis celui de la boucle principale selon la figure suivante :



La synthèse de PI de la boucle interne :

La boucle interne de ce système est la même que celle du système pompe-débit en boucle fermé, alors on garde le même régulateur PI qu'on a synthétisé.

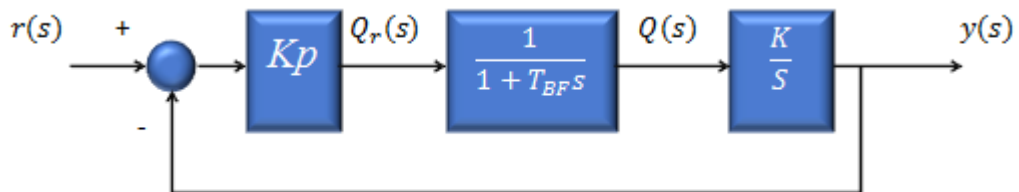
$$R(s) = 1.9482 \left(1 + \frac{1}{2.7s} \right) \quad (\text{IV. 119})$$

La fonction du transfert de la boucle interne est donc donnée par l'expression suivante :

$$Ty(s) = \frac{1}{1+T_{BFS}} = \frac{1}{1+2.16s} \quad (\text{IV. 120})$$

La synthèse du gain proportionnel de la boucle principale (Kp):

Puisque le système possède déjà un intégrateur. Alors il suffit d'utiliser une action proportionnelle.



La fonction du transfert en boucle ouverte est donnée par la relation suivante :

$$G(s) = \frac{K_p K}{s(1+T_{BFS})} = \frac{0.08K_p}{s(1+2.16s)} \quad (\text{IV. 121})$$

La fonction de transfert en boucle fermé est donnée par la relation suivante :

$$G_{bf}(s) = \frac{G(s)}{1+G(s)} = \frac{KK_p}{T_{BFS}s^2 + s + KK_p} \quad (\text{IV. 122})$$

$$G_{bf}(s) = \frac{1}{\frac{1}{\omega_n^2}s^2 + \frac{2\xi}{\omega_n}s + 1} \quad (\text{IV. 123})$$

Par identification entre (IV.114) et (IV.115) on trouve :

$$\frac{1}{\omega_n^2} = \frac{T_{BF}}{KK_P} \qquad \frac{2\varepsilon}{\omega_n} = \frac{1}{KK_P} \qquad (IV. 124)$$

D'après les deux équations trouvées précédemment on a :

$$\omega_n = \frac{1}{2\varepsilon T_{BF}} ; \qquad K_P = \frac{\omega_n}{2\varepsilon K} \qquad (IV. 125)$$

En imposant $\varepsilon = 0.7$ pour avoir une meilleure réponse, on trouve :

$$\omega_n = 0.3306 \qquad K_P = 2.9517 \qquad (IV. 126)$$

La fonction du transfert du régulateur est donnée par l'expression suivante:

$$R(s) = K_p = 2.9517$$

(IV. 127)

IV.B.3.Régulation de pression

a. Système pompe-pression :

a.1.Régulation par PI

Pour avoir un comportement du premier ordre en boucle fermé, on synthétise le régulateur PI par la méthode de compensation.

Soit la fonction de transfert en boucle ouverte du système pompe-pression et compris le capteur :

$$H(s) = \frac{0.6125}{12s+1} \qquad (IV. 128)$$

Comme le capteur à un gain de 0.01, alors pour obtenir le modèle réel du système (sans le transfert du capteur) il suffit de multiplier H(s) par 100.

On obtient la fonction de transfert en boucle ouverte du système bouclé suivante :

$$G(s) = R(s) * H(s) = \left[K_p \left(1 + \frac{1}{T_i s} \right) \right] \left[\frac{K}{\tau s + 1} \right] = \left[K_p \left(1 + \frac{1}{T_i s} \right) \right] \left[\frac{61.25}{12s+1} \right] \qquad (IV. 129)$$

Pour avoir un comportement en premier ordre en boucle fermé on prend $T_i = \tau$:

$$T_i = \tau \Rightarrow G(s) = \frac{K_p K}{\tau s} = \frac{a}{s} \Rightarrow G_{bf}(s) = \frac{G(s)}{1+G(s)} = \frac{a}{s+a} \qquad (IV. 130)$$

$$(IV. 131) \Rightarrow \frac{1}{\frac{1}{a}s+1} = \frac{1}{T_0s+1} \quad (IV. 132)$$

On prend $T_0 = \frac{\tau}{2}$ pour accélérer un petit peu le système en boucle fermé :

$$T_0 = \frac{\tau}{2} \Rightarrow \frac{\tau}{K_p K} = \frac{\tau}{2} \Rightarrow K_p = \frac{2}{K} = 0.0326 \quad (IV. 133)$$

Donc le régulateur PI vaut :

$$R(s) = 0.0326 \left(1 + \frac{1}{12s} \right) \quad (IV. 134)$$

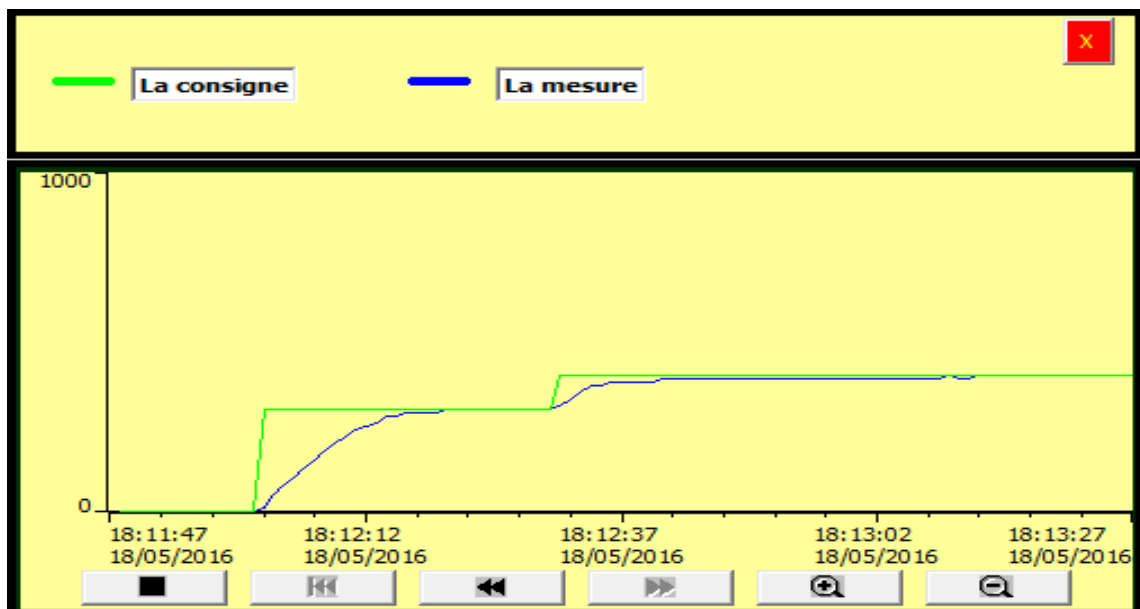


Figure IV.16: la réponse du système pompe-pression commandé par un PI sous WinCC

a.2. Régulation par mode de glissement

Pour synthétiser le régulateur par le mode de glissement, on a utilisé le modèle d'état obtenu dans le chapitre précédent :

$$\begin{cases} \dot{x} = -0.0833x + 5.1041u \\ y = x \end{cases} \quad (IV. 135)$$

On choisit premièrement la surface sur laquelle les objectifs de commande sont satisfaits, pour cela on prend la surface de Slotine et Li : $S = (\gamma + \frac{d}{dt})^{r-1} e$; $\gamma > 0$

On prend $e = x - x_r$, on obtient : $S = e$.

Ensuite, on cherche la commande pour laquelle on assure l'attractivité de la surface et le maintien ; il faut donc poser :

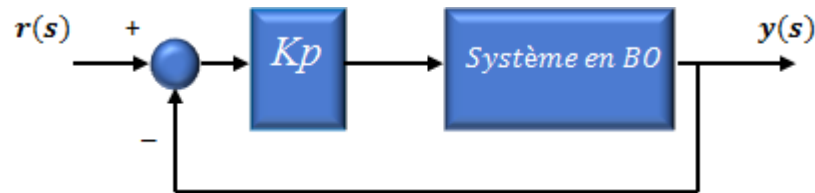
$$\dot{s} = -K \operatorname{sign}(s) \Rightarrow -0.0833x + 5.1041u = -K \operatorname{sign}(s) \quad (\text{IV. 136})$$

$$u = 0.0163x - 0.1959 K \operatorname{sign}(x - x_r) \quad (\text{IV. 137})$$

b. Système vanne-pression

b.1. Régulation par un gain proportionnel

Le système possède déjà un intégrateur c'est pour cela on utilise que l'action proportionnelle.



La fonction de transfert en boucle ouverte du système vanne-pression :

$$H(s) = \frac{0.027}{s} \quad (\text{IV. 138})$$

Comme le capteur à un gain de 0.01, alors pour obtenir le modèle réel du système (sans le capteur) il suffit de multiplier $H(s)$ par 100.

Donc la fonction de transfert en boucle ouverte du système de régulation est donné par :

$$G(s) = \frac{KK_P}{s} \text{ avec } K = 2.7 \quad (\text{IV. 139})$$

La fonction de transfert en boucle fermée est donnée par la relation suivante :

$$G_{bf}(s) = \frac{G(s)}{1+G(s)} = \frac{KK_P}{s+KK_P} \Rightarrow \frac{1}{\frac{1}{KK_P}s+1} = \frac{1}{T_0s+1} \quad (\text{IV. 140})$$

$$\text{On impose } T_0 = 2 \Rightarrow Kp = \frac{1}{KT_0} \quad (\text{IV. 141})$$

La fonction du transfert du régulateur est donnée par l'expression suivante:

$$R(s) = K_p = 0.1851$$

(IV. 142)

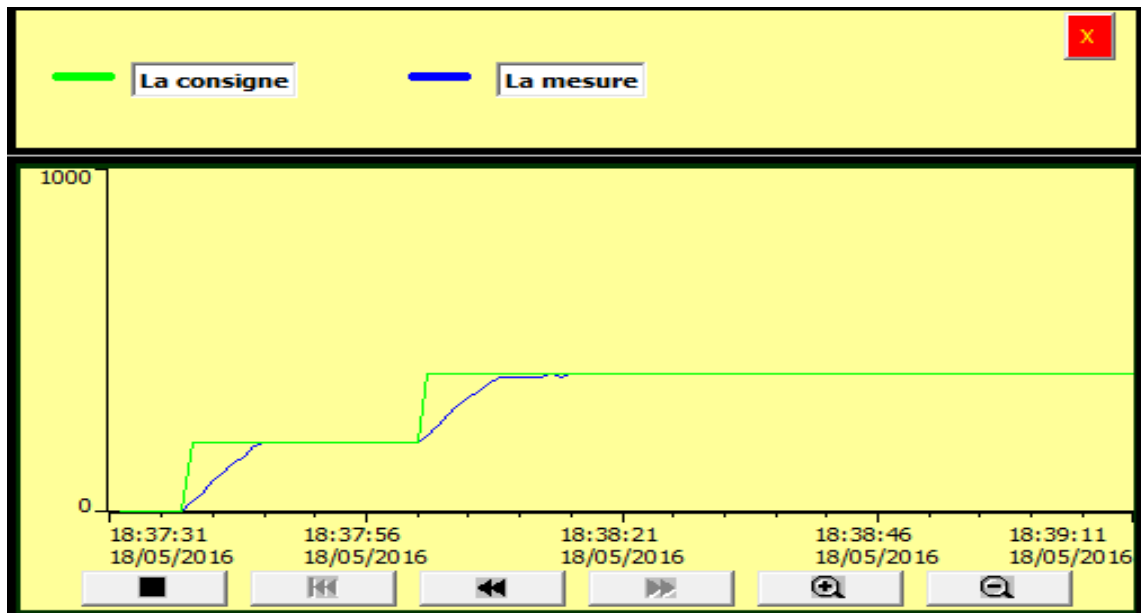


Figure IV.17: la réponse du système vanne-pression commandé par un gain sous WINCC

b.2.Régulation par mode de glissement

Pour synthétiser le régulateur par la méthode de mode de glissement, on utilise le modèle d'état qu'on a présenté dans le chapitre précédent :

$$\begin{cases} \dot{x} = 2.7u \\ y = x \end{cases} \quad (\text{IV. 143})$$

On choisit premièrement la surface sur laquelle les objectifs de commande sont satisfaits, pour cela on prend la surface de Slotine et Li : $S = (\gamma + \frac{d}{dt})^{r-1}e$; $\gamma > 0$.

On prend $e = x - x_r$ et on obtient par la suite $S = e$

Ensuite, on cherche la commande pour laquelle on assure l'attractivité de la surface et le maintien ; il faut donc poser $\dot{s} = -K \text{sing}(s) \Rightarrow 2.7 u = -K \text{sing}(s)$ (IV. 144)

$$u = -0.3703 K \text{sign}(x - x_r) \quad (\text{IV. 145})$$

La figure suivante représente la réponse obtenue après l'application de la commande par mode de glissement.

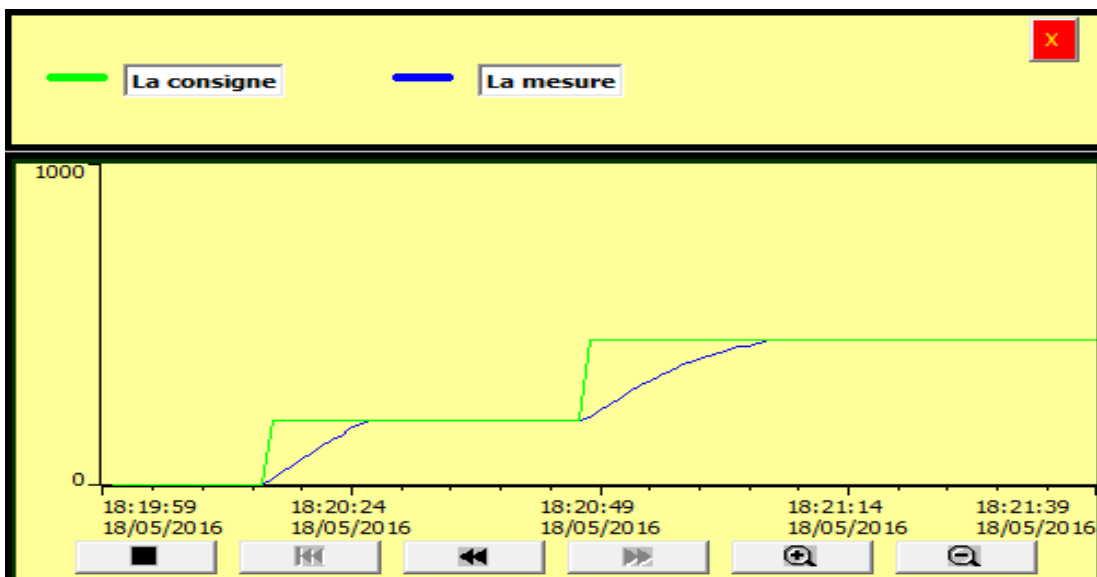


Figure IV.18: la réponse du système vanne-pression commandé par mode glissant sous WinCC

b.3. Commande discrète :

Pour cette commande, on a élaboré un algorithme qui adapte la commande avec l'erreur par plage. L'organigramme de commande discrète du système vanne-pression est le suivant :

x_r : la consigne x : la mesure e : l'erreur de régulation

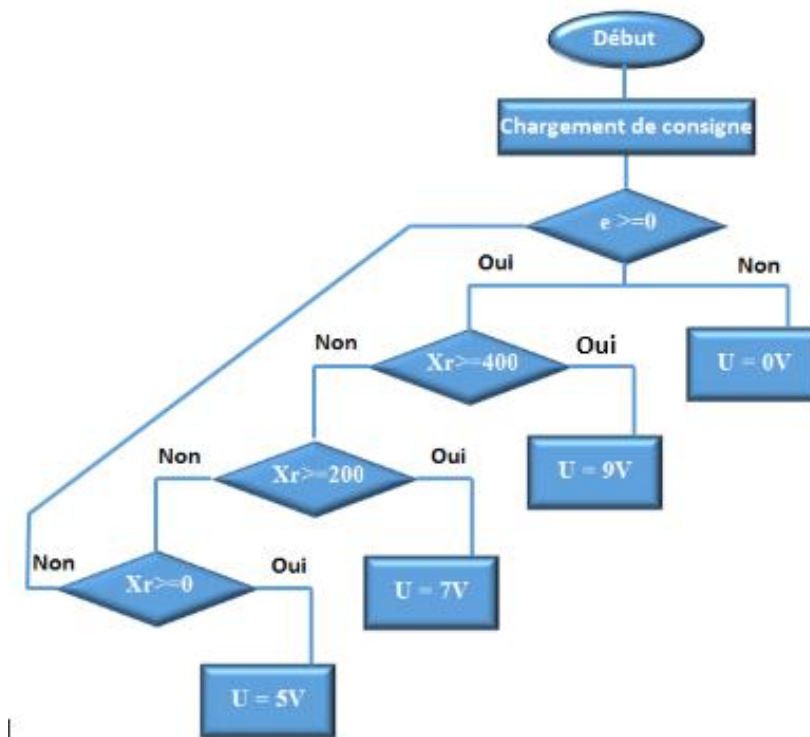


Figure IV. 19: L'organigramme de commande discrète du système vanne-pression

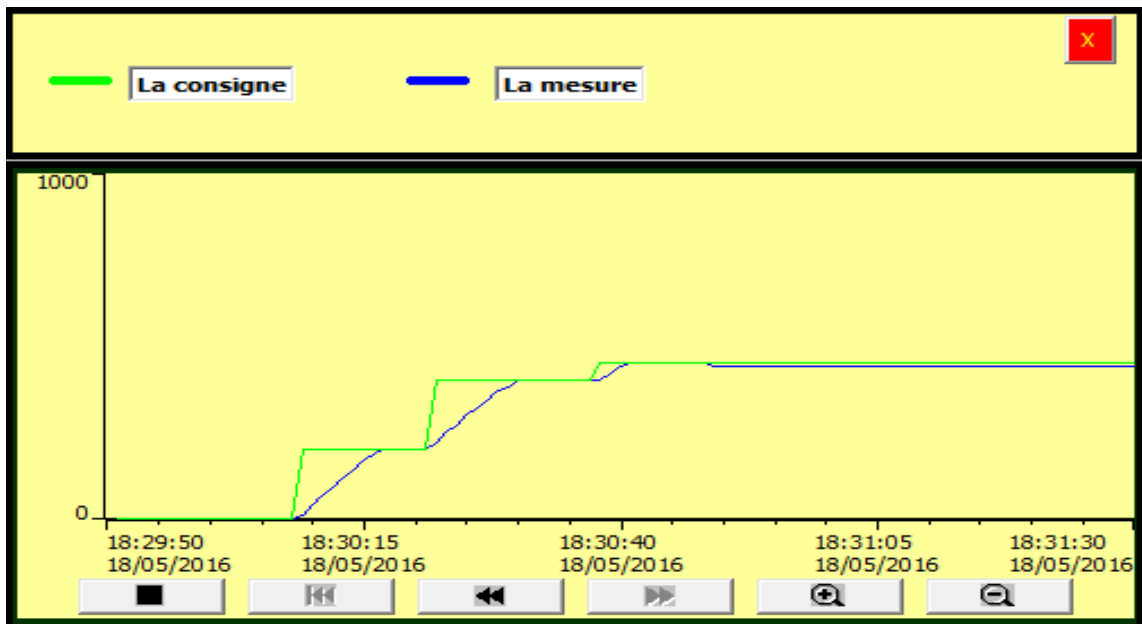


Figure IV.20: la réponse du système vanne-pression commandé par la commande discrète sous WINCC

5. Problèmes de variations paramétriques et des erreurs de modélisations :

Toutes les grandeurs (débit, niveau, pression) peuvent être réglées par l'un des deux actionneurs (pompe ou vanne). Alors lors de l'identification, on a fixé la commande de l'un des deux à une valeur précise, et on a identifié le système en prenant la tension de deuxième actionneur comme commande. Donc si cette valeur change alors le modèle du système change et ça va engendrer des variations paramétriques.

On prend l'exemple du système pompe-débit, si le degré d'ouverture de la vanne change, ceci va engendrer des variations paramétriques qui vont changer le modèle du système. Donc pour la résolution de ce problème, on va utiliser une commande adaptative indirecte qui est une approche de la commande robuste et puis une adaptation des paramètres par la logique floue.

a. commande adaptative indirecte :

Le système en boucle ouverte a le modèle d'état suivant:

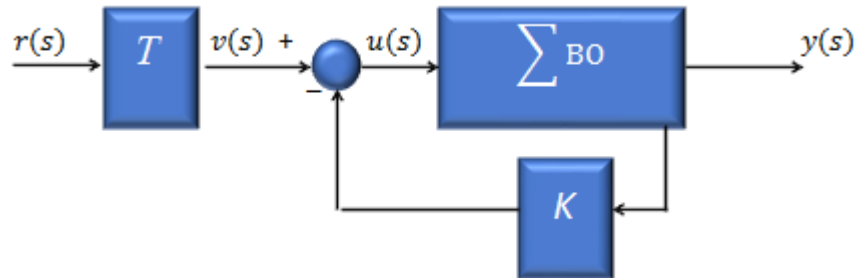
$$\begin{cases} \dot{x} = -ax + bu \\ y = x \end{cases} \quad (\text{IV. 146})$$

θ : vecteur des paramètres, il est inconnue.

$$\theta = \begin{pmatrix} a \\ b \end{pmatrix} \quad (\text{IV. 147})$$

1. La synthèse de la commande :

On suppose qu'on connaît le vecteur des paramètres du système et on synthétise la commande en utilisant un retour d'état plus un pré-compensateur.



- Le retour d'état :

Le modèle d'état du système est donné sous la forme suivante :

$$\dot{x} = -ax + bu \quad \text{avec : } a > 0$$

Pour le retour d'état on prend :

$$u = -Kx \quad \Rightarrow \quad \dot{x} = (-a - bK)x + bv \Rightarrow \dot{x} = a_{bf}x + bv \quad (\text{IV. 148})$$

a_{bf} est le pôle de la boucle fermé, donc pour accélérer le système on prend : $a_{bf} = -2a$

$$(-a - bK) = -2a \Rightarrow K = \frac{a}{b} \quad (\text{IV. 149})$$

- Pré-compensateur :

Le retour d'état n'assure pas la poursuite donc on est obligé de rajouter un pré-compensateur :

$$\dot{x} = a_{bf}x + bv \quad \Rightarrow \quad s x(s) = a_{bf}x(s) + b v(s) \quad (\text{IV. 150})$$

$$\Rightarrow (s - a_{bf})x(s) = b v(s) \quad (\text{IV. 151})$$

$$\Rightarrow x(s) = \frac{b}{s - a_{bf}} v(s) \quad (\text{IV. 152})$$

Et comme $v(s) = T.r(s)$ (IV.144) alors :

$$x(s) = \frac{bT}{s - a_{bf}} r(s) \quad (\text{IV. 1535})$$

On a:

$$\begin{cases} y(s) = x(s) \\ x(s) = \frac{bT}{s - a_{bf}} r(s) \end{cases} \Rightarrow y(s) = \frac{bT}{s - a_{bf}} r(s) \quad (\text{IV. 154})$$

Pour une bonne poursuite de la référence en régime statique on prend :

$$T = \frac{1}{b} \quad (\text{IV. 155})$$

D'après l'équation (IV.140) et (IV. 156) on trouve :

$$\text{On prend : } X_r = r \quad (\text{IV. 157})$$

$$u = -K.X + T.X_r \Rightarrow u = \left(-\left(\frac{a}{b}\right) * X\right) + \left(\left(\frac{1}{b}\right) * X_r\right) \quad (\text{IV. 158})$$

5. La synthèse de l'algorithme d'adaptation paramétrique (AAP) :

On choisit le signal y_i :

$$y_i = \varphi^T \theta(t) = \dot{x} = -ax + bu = (-x \quad u) \begin{pmatrix} a \\ b \end{pmatrix} \quad \text{Avec } \varphi : \text{ est le vecteur de mesure.} \quad (\text{IV. 159})$$

d- On construit le prédicteur de y_i noté \hat{y}_i avec :

$$\begin{cases} \hat{y}_i(t) = \varphi^T \hat{\theta}(t) \\ \hat{\theta}(t): \text{ l'estimé à l'instant } t \text{ de } \theta(t) \end{cases} \quad (\text{IV. 160})$$

Alors l'erreur de prédiction sera:

$$e = y_i - \hat{y}_i(t) = \varphi^T(\theta(t) - \hat{\theta}(t)) \quad (\text{IV. 161})$$

e- le critère J à optimiser :

$$J = \frac{1}{2} e^2 \quad (\text{IV. 162})$$

La descente de gradient :

$$\dot{\hat{\theta}} = -\gamma \left(\frac{dJ}{d\hat{\theta}}\right)^T \quad \text{Avec } \gamma > 0 \quad (\text{IV. 163})$$

$$(\text{IV. 164}) \Rightarrow \dot{\hat{\theta}} = \gamma \cdot e \cdot \varphi^2 \quad (\text{IV. 165})$$

On fait une approximation du dérivé par une dérivation numérique :

$$\dot{\hat{\theta}} = \frac{\hat{\theta}(k+1) - \hat{\theta}(k)}{\Delta} \quad \text{Avec } \Delta: \text{ pas d'échantionnage.} \quad (\text{IV. 166})$$

D'après l'équation (IV.156) et (IV.155) on formule l'algorithme d'adaptation paramétrique suivant:

$$\hat{\theta}(k+1) = \hat{\theta}(k) + \Delta \cdot \gamma \cdot e \cdot \varphi \quad \Rightarrow \quad \begin{cases} a(k+1) = a(k) - \Delta \cdot \gamma \cdot 1 \cdot e \cdot x \\ b(k+1) = b(k) + \Delta \cdot \gamma \cdot 2 \cdot e \cdot u \end{cases} \quad (\text{IV. 167})$$

6. La troisième étape consiste à remplacer le vecteur de paramètres par son estimateur dans l'expression de la commande synthétisé précédemment.

$$u = \left(- \left(\frac{a(k)}{b(k)} \right) * X \right) + \left(\left(\frac{1}{b(k)} \right) * X_r \right)$$

(IV. 168)

7. L'implémentation de l'algorithme:

- On démarre l'algorithme toujours par les valeurs nominales :

$$\begin{cases} a(k) = a_0 = 0.3377 \\ b(k) = b_0 = 0.2050 \end{cases}$$

(IV. 169)

En choisissant les valeurs suivantes, on a obtenu la réponse ci-dessous :

$\Delta = 0.05(s)$: pas d'échantillonnage

$\gamma_1 = \gamma_2 = 0.01$: paramètres de convergence

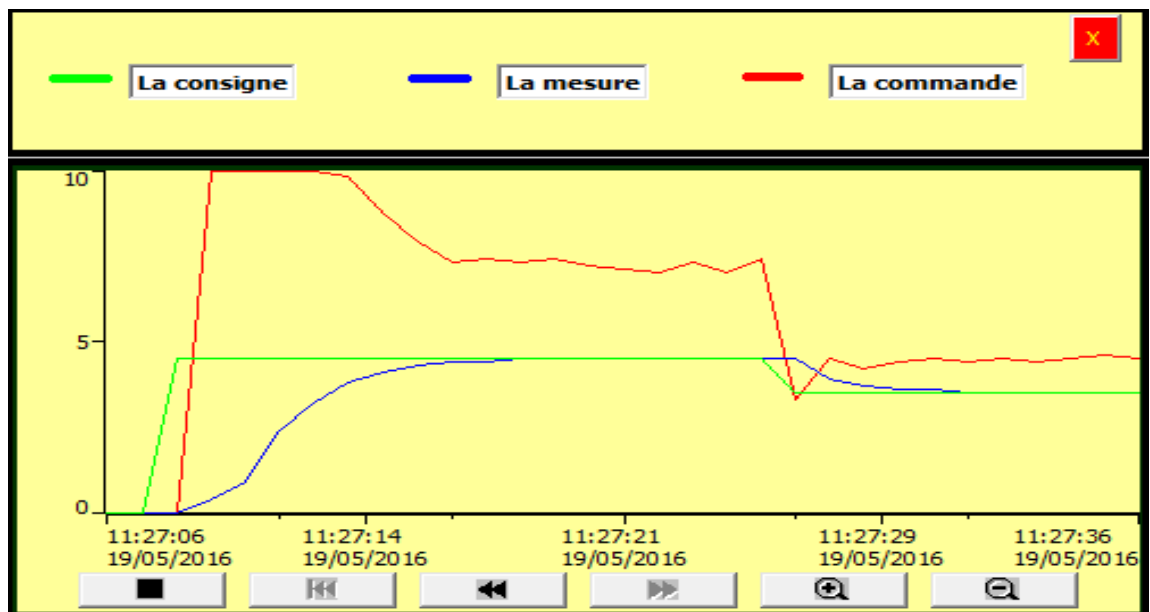


Figure IV.21 : La réponse du système pompe-débit sous WinCC

b. Régulateur PI avec adaptation paramétrique par la logique floue :

Si on joue sur l'ouverture de la vanne proportionnelle, le gain du système pompe-débit change, donc on aura des variations paramétriques et le modèle identifier n'est plus le même. Pour résoudre ce problème, on va faire une adaptation par logique floue du gain du système, selon les étapes suivant :

Fuzzification: On transforme l'angle d'ouverture de la vanne " α " en une variable floue α_F qui peut prendre les valeurs floues suivantes : {Trop petit « TP », Petit « P », Moyen «M», Grand « G » et Très Grand « TG »} avec des fonctions d'appartenance suivante la figure ci-dessous :

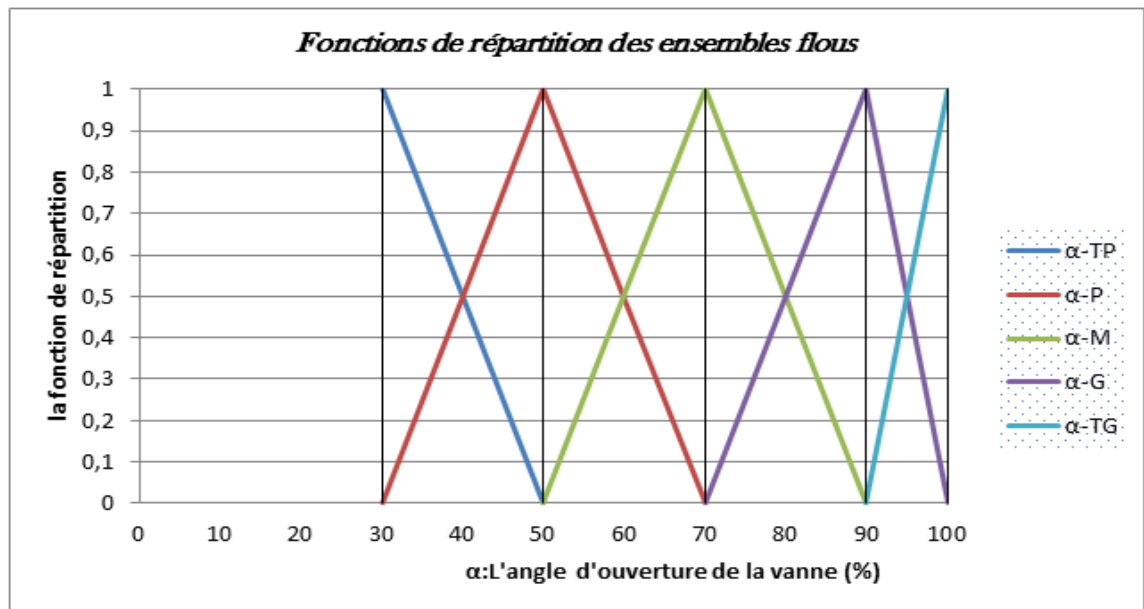


Figure IV.22: les fonctions d'appartenance de la variable floue α_F

1- On établit les règles sous l'approche de TSK :

Règle 1 : si α_F est TP alors $K_1 = 0.5000$

Règle 2 : si α_F est P alors $K_2 = 0.5343$

Règle 3 : si α_F est M alors $K_3 = 0.5775$

Règle 4 : si α_F est G alors $K_4 = 0.5900$

Règle 5 : si α_F est TG alors $K_5 = 0.5993$

Soit α_i : le degré d'activation de la règle i.

$$\text{Règle 1} \rightarrow \alpha_1 = \mu_{TP}(y_F)$$

$$\text{Règle 2} \rightarrow \alpha_2 = \mu_P(y_F)$$

$$\text{Règle 3} \rightarrow \alpha_3 = \mu_M(y_F)$$

$$\text{Règle 4} \rightarrow \alpha_4 = \mu_G(y_F)$$

$$\text{Règle 5} \rightarrow \alpha_5 = \mu_{TG}(y_F)$$

D'après l'approche de TSK, le gain réel du système est donné par :

$$K = \frac{\sum \alpha_i * K_i}{\sum \alpha_i} \quad (\text{IV. 170})$$

La synthèse du régulateur PI :

$$R(s) = K_p \left(1 + \frac{1}{2.9626 s} \right) \text{ Avec : } K_p = \frac{1.25}{K}$$

(IV. 171)

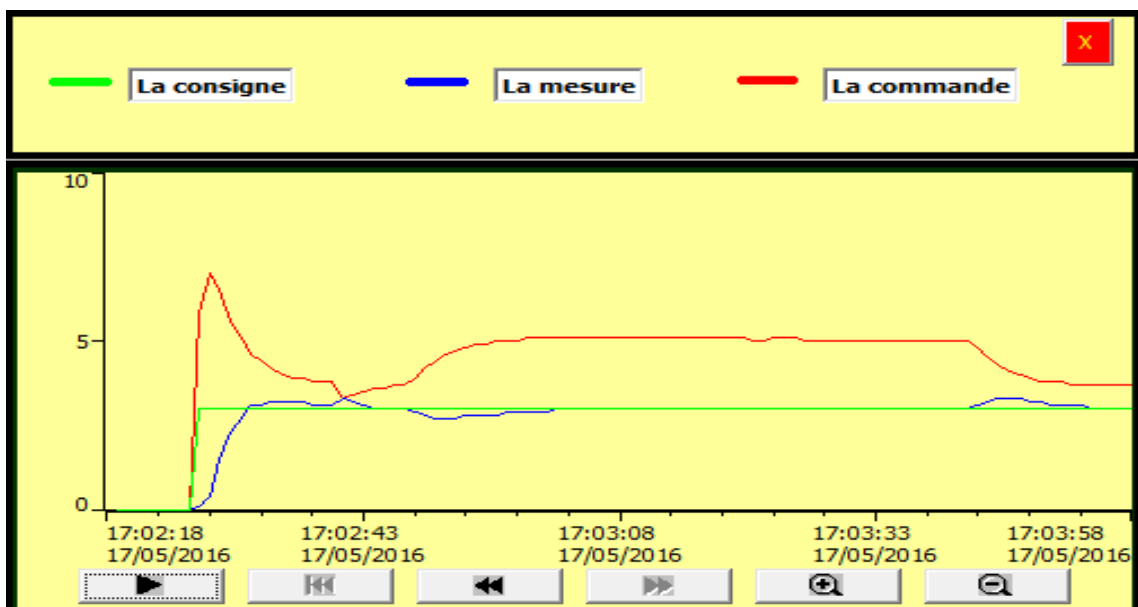


Figure IV.23 : La réponse du système pompe-débit sous WinCC

IV.B.4. Régulation de température

Pour régler la température du fluide de la cuve agitée, on va utiliser le circuit fluide chaud. Dans ce circuit la pompe P1 conduit l'eau chaude stockée dans la cuve de chauffe au serpentin situé dans la cuve agitée, là il va y avoir un échange thermique avec le fluide de la cuve agitée à travers les parois du serpentin.

Dans ce circuit la résistance de chauffe située à l'intérieur de la cuve de chauffe va chauffer l'eau stockée dans cette dernière. Donc la commande de ce système va être la tension d'entrée de la pompe P1.

Puis que le système de réglage de la température est très lent on va utiliser une commande tout ou rien.

L'algorithme de régulation est le suivant

x_r : la consigne x : la mesure e : l'erreur de régulation u : la commande

$$e = x_r - x$$

$$\text{Si } e > 5^\circ\text{C} \text{ alors : } u = 9\text{V}$$

$$\text{Si non si } 5^\circ\text{C} \leq e < 0^\circ\text{C} \text{ alors : } u = 4\text{V}$$

$$\text{Si non } u = 0\text{V}$$

PARTIE C : Implémentation et supervision des différentes processus d'automatisation.

IV.C.1 Implémentation sur Step7 :

Pour pouvoir régler toutes les systèmes qu'on a identifiés, il faut d'abord lire les valeurs réelles des grandeurs à régler (Débit, Niveau, Pression, Température) à travers les capteurs, puis après le calcul de la commande dans l'automate on envoie la commande directement vers l'actionneur à travers le module analogique.

Pour faire ceci il faut faire un adressage pour chaque entrée/sortie au niveau de la table des mnémonique et faire une mise à l'échelle des entrées et des sorties au niveau du programme principale OB1.

Adressage des entrées/sorties :

PEW	128	Débits
PEW	130	Niveau
PEW	132	Pression
PEW	134	Température dans la cuve
PAW	128	l'entrée de l'actionneur

La mise à l'échelle :

Pour la mise à l'échelle des entrées qui viennent des capteurs et la sortie qui se dirige vers les actionneurs, on a utilisé les deux fonctions qui se trouvent dans la bibliothèque du Step7 : FC105 (SCALE) et FC106 (UNSCALE) respectivement.

1- La fonction SCALE (FC105):

La fonction Mise à l'échelle (SCALE) prend une valeur entière (IN) et la convertit selon l'équation ci-après en une valeur réelle exprimée en unités physiques, comprises entre une limite inférieure (LO_LIM) et une limite supérieure (HI_LIM) :

$$OUT = [((FLOAT (IN) - K1) / (K2 - K1)) * (HI_LIM - LO_LIM)] + LO_LIM$$

Le résultat est écrit dans OUT.

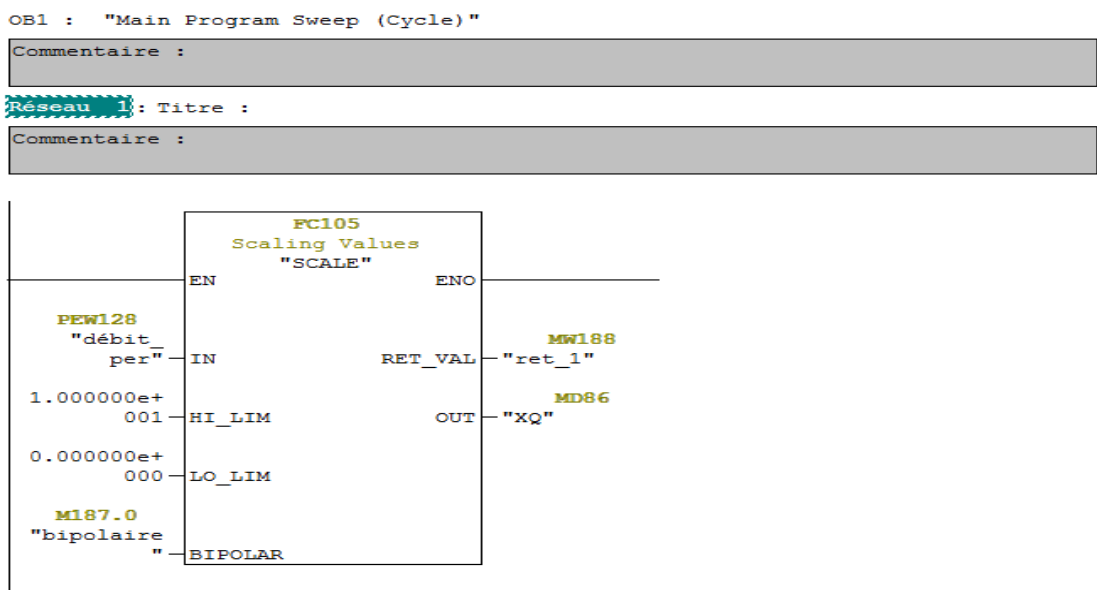


Figure IV.24: Le bloc de la fonction SCALE

Les constantes K1 et K2 sont définies selon que la valeur d'entrée est bipolaire ou unipolaire.

- Bipolaire : K1 = -27648.0 et K2 = +27648.0
- Unipolaire : K1 = 0.0 et K2 = +27648.0

Si la valeur entière d'entrée est supérieure à K2, la sortie (OUT) est saturée à la valeur la plus proche de la limite supérieure (HI_LIM) et une erreur est signalée. Si la valeur entière d'entrée est inférieure à K1, la sortie est saturée à la valeur la plus proche de la limite inférieure (LO_LIM) et une erreur est signalée.

2- La fonction UNSCALE (FC106):

La fonction Annuler la mise à l'échelle (UNSCALE) prend une valeur d'entrée réelle (IN) exprimée en unités physiques comprises entre une limite inférieure (LO_LIM) et une limite supérieure (HI_LIM) et la convertit selon l'équation ci-après en une valeur entière :

$$OUT = [((IN-LO_LIM)/(HI_LIM-LO_LIM)) * (K2-K1)] + K1$$

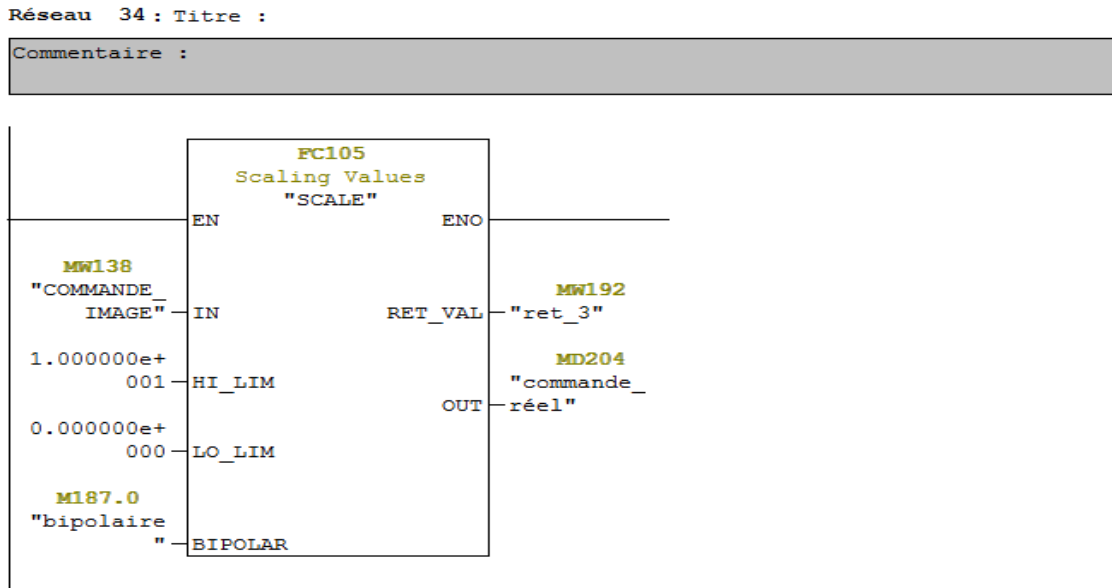


Figure IV.25: Le bloc de la fonction UNSCALE

Pour l'implémentation des différentes commandes qu'on a synthétisées, on a créé des blocs fonctionnels à partir des fichiers sources en utilisant le langage structuré, et par la suite on a fait appel à ces blocs dans le programme principale OB1.

- Les étapes de l'implémentation des commandes:

1- Création d'un fichier source:

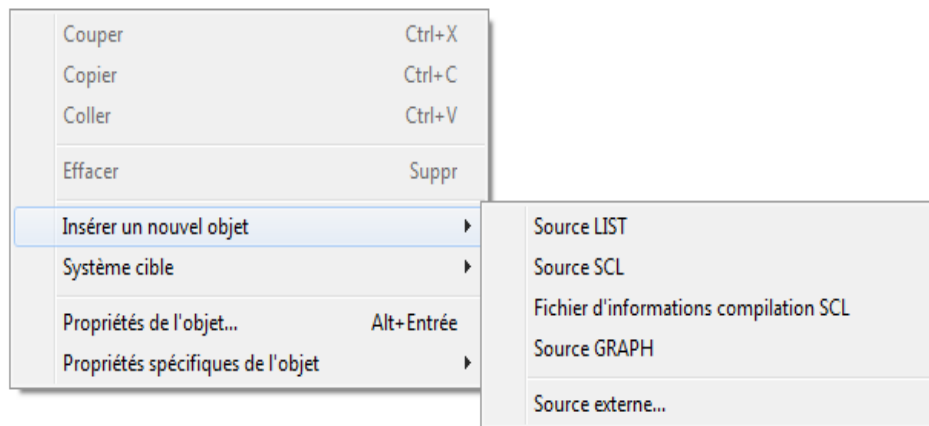


Figure IV.26: La création d'un fichier source

2- Le choix du modèle de bloc :

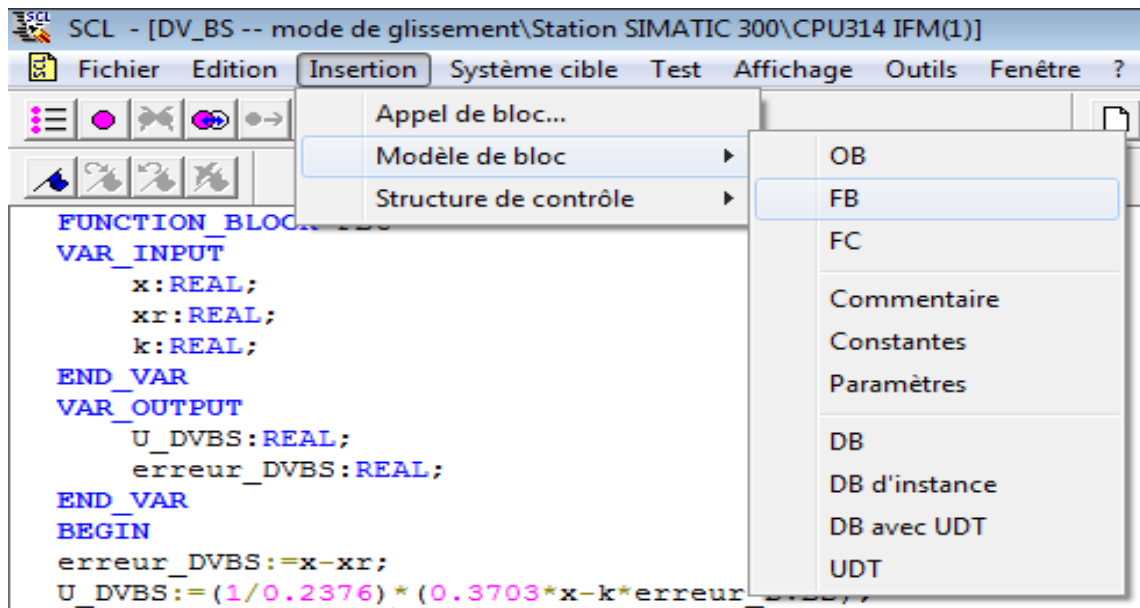


Figure IV.27: Le choix du modèle de bloc

3- La déclaration des variables et la saisie du programme:

Pour la programmation des blocs fonctionnels on a utilisé le langage structuré comme montre la figure:

```

FUNCTION_BLOCK FB2
VAR_INPUT
    x:REAL;
    xr:REAL;
    k:REAL;
END_VAR
VAR_OUTPUT
    U_DPMG:REAL;
    erreur_DPMG:REAL;
END_VAR
BEGIN
    erreur_DPMG:=x-xr;
    U_DPMG:=(1/0.2050)*(0.3377*x-k*ATAN(erreur_DPMG));
END_FUNCTION_BLOCK

```

Figure IV.28: Programme sous le langage structuré dans un fichier source

4- La compilation du programme:

La compilation sert à détecter les erreurs, et à générer le bloc fonctionnel qui correspond au programme compilé.

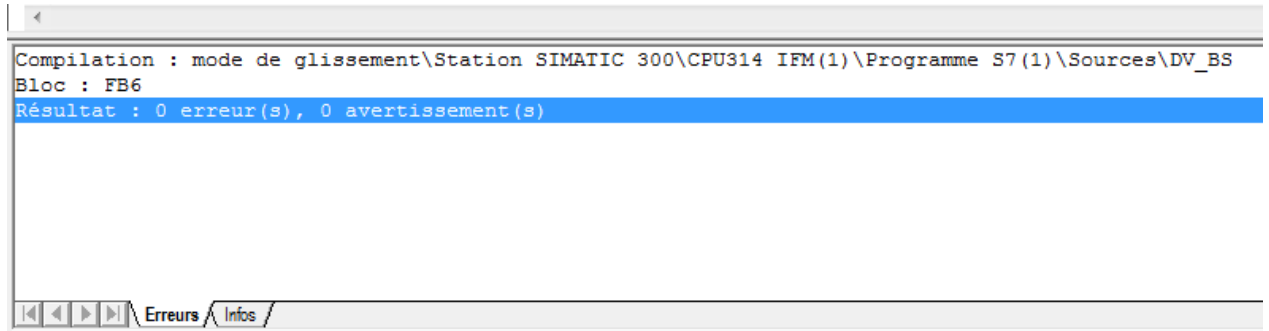


Figure IV.29: La fenêtre des erreurs de compilation

5- L'appel du bloc dans le programme principal OB1:

Pour appeler ce bloc dans le OB1 il suffit d'aller sur l'icône FB et de choisir le bloc fonctionnel et de le glisser à l'intérieure du réseau.

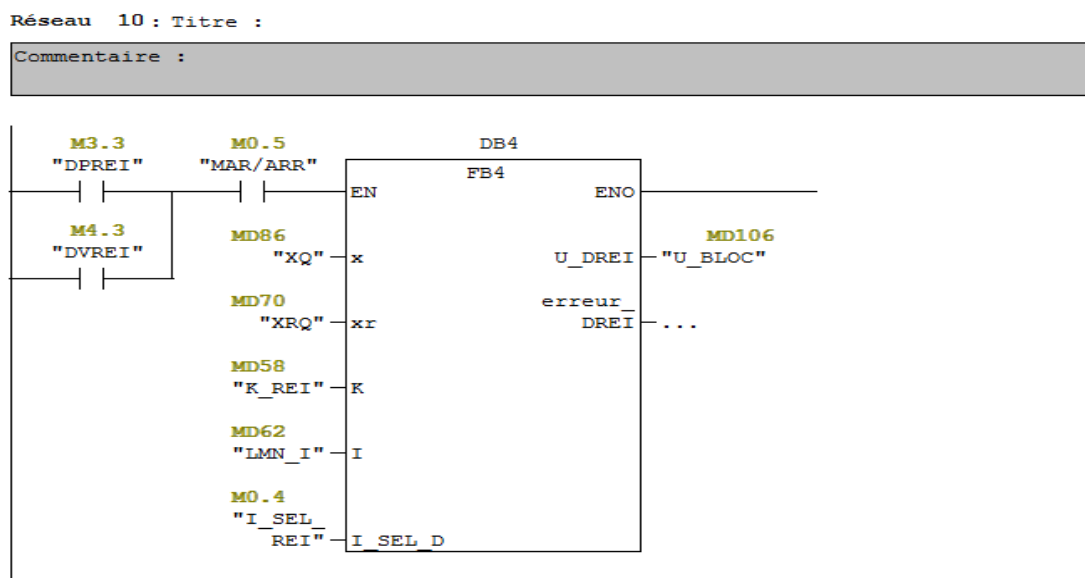


Figure IV.30: Bloc de régulation appelé dans un OB1

6- La création d'un bloc de donné qui correspond au bloc fonctionnel:

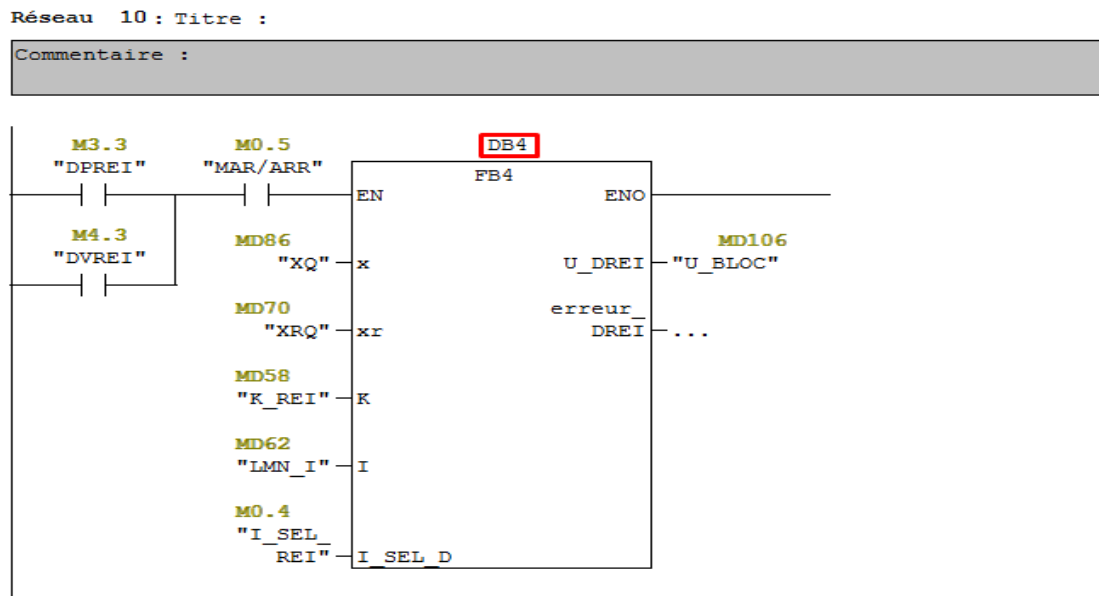


Figure IV.31: création du bloc de donné

Certains blocs qui contiennent une dérivation ou une intégration numérique (nécessitent un temps d'échantionnage précis) comme le FB41, on les appelle dans des OB d'alarme cyclique.

La CPU S7 314 IFM met à notre disposition l'OB35 qui interrompt le traitement de programme cyclique à intervalles précis. Dans la configuration matérielle, il est possible de choisir la fréquence d'exécution de l'OB35 en allant dans les propriétés de la CPU sous l'onglet Alarmes cycliques. Ce temps ne doit pas être trop court. On doit s'assurer que le reste du programme et notamment l'OB1 ait le temps de s'exécuter entre deux appels de l'OB35.

Les alarmes cycliques sont déclenchées à des intervalles de temps précis. Le moment de déclenchement de la période est le passage de l'état de fonctionnement "Arrêt" (STOP) à l'état "Marche" (RUN). Le tableau suivant montre les périodes et classes de priorité prédéfinies des OB d'alarme cyclique. Vous pouvez modifier les périodes et classes de priorité par paramétrage.

OB d'alarme cyclique	Période (Ms)	Classe de priorité
OB30	5000	7
OB31	2000	8
OB32	1000	9
OB33	500	10
OB34	200	11
OB35	100	12
OB36	50	13
OB37	20	14
OB38	10	15

Tableau IV.1: les OB d'alarme cyclique

IV.C.2 Supervision sur WinCC :

Pour une bonne visualisation et pour contrôler facilement les opérations d'automatisations, on a créé une interface graphique sur le WinCC.

Cette interface est composée de deux vue, la vue principale et la vue secondaire.

Dans la vue principale de l'interface graphique, on trouve :

- 1- un schéma représentatif de la station CE117.....(1)
- 2- Une fenêtre partagée en 3 parties :
 - La première partie c'est pour l'état du système (Marche / Arrêt)(2)
 - La deuxième partie c'est pour le choix des paramètres de réglage(3)

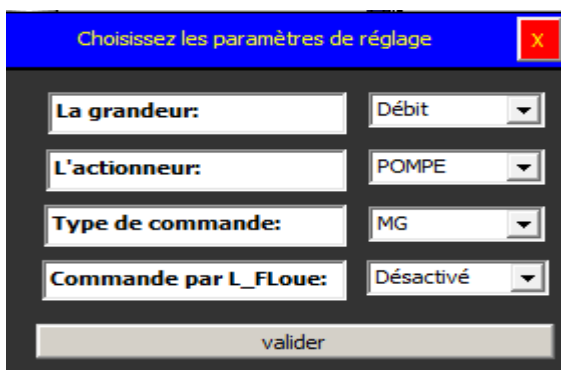


Figure IV.32 : La fenêtre de choix des paramètres de réglage

- La troisième partie est consacrée au contrôle de système(4)

Dans la deuxième partie, on a les possibilités suivantes :

- Le choix de système de réglage : la grandeur, l'actionneur et le type de commande.
- La saisie des paramètres de commande(5)
- La visualisation de la courbe de réglage(6)
- 3- Un bouton qui permet le choix entre la commande automatique et manuelle..... (7)
- 4- Une fenêtre d'alarme pour l'affichage des alarmes et des avertissements.....(8)
- 5- Un tableau d'affichage de système de réglage..... (9)

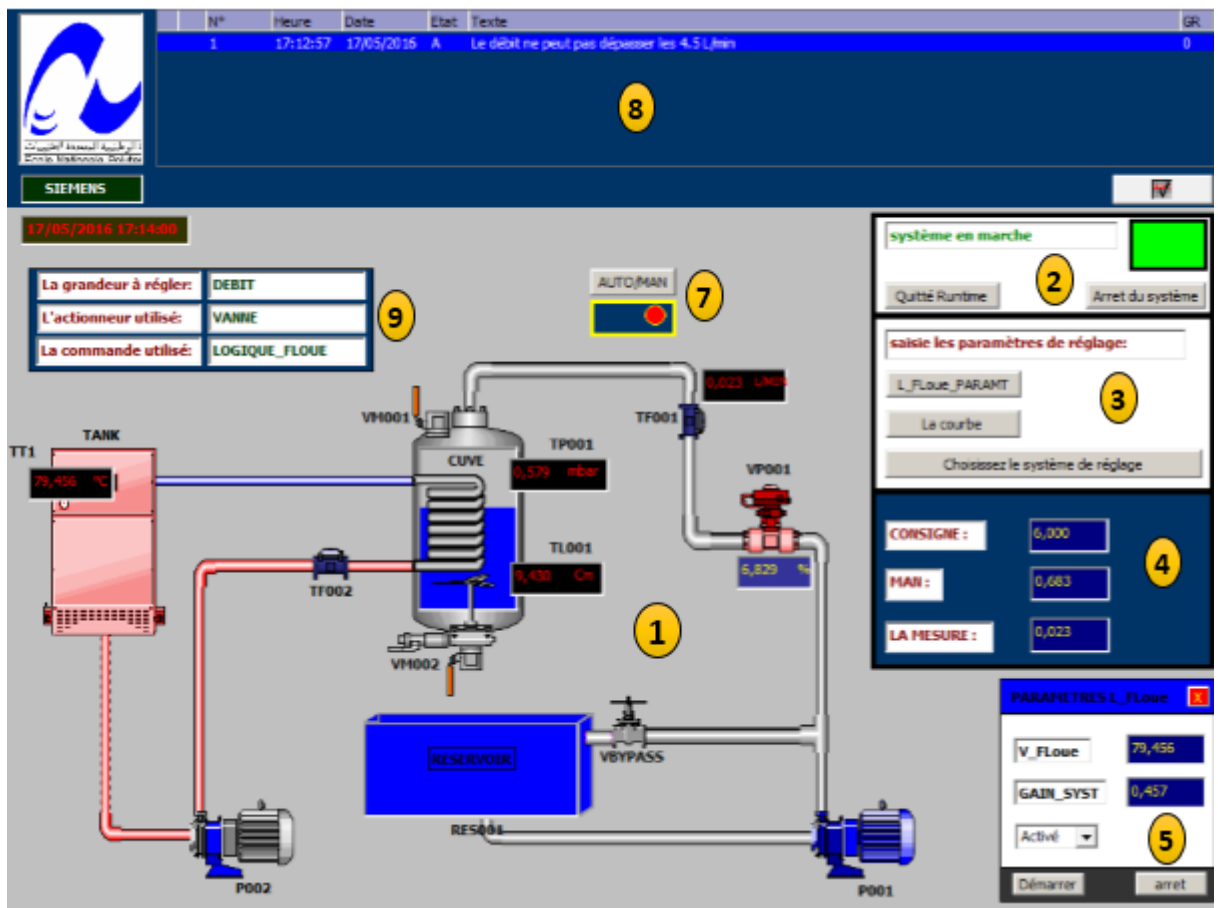


Figure IV.33 : La vue principale de l'interface graphique IHM

Dans la vue secondaire de l'interface graphique, on trouve :

- La visualisation de la courbe de réglage.....(6)
- Fenêtre de contrôle l'état de système (Marche / Arrêt)(10)
- Fenêtre pour la saisie de la consigne et pour visualiser la mesure(11)
- Fenêtre pour la saisie des paramètres de commande (12)

- Une fenêtre d'alarme pour l'affichage des alarmes et des avertissements.....(8)

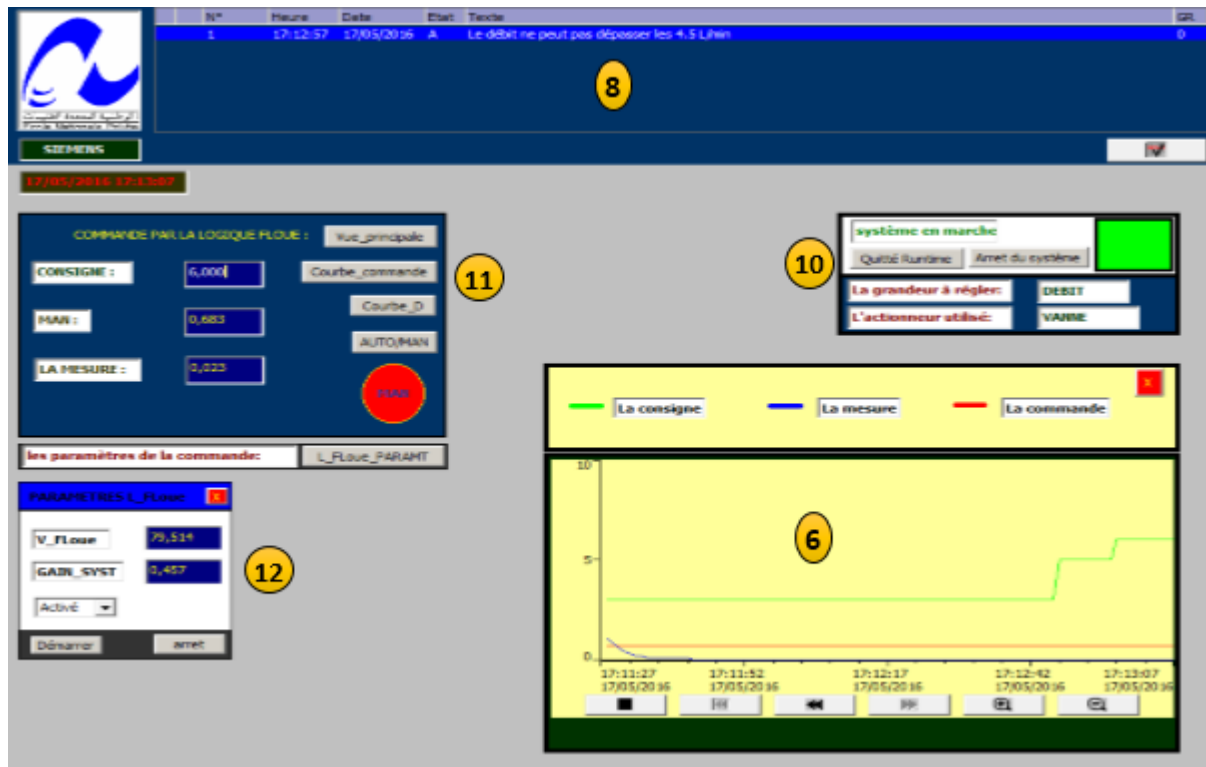


Figure IV. 34: La vue secondaire de l'interface graphique IHM

Pour bien comprendre comment faire en détail l'implémentation des différentes commandes utilisées sur STEP7 et la supervision sur WinCC, on a créé des vidéos pédagogiques qu'on a publiées sur le site web YouTube selon les liens suivants :

- 1- Comment faire une régulation par le logiciel STEP7 ? (ex. régulation de pression par la commande de PI) https://www.youtube.com/watch?v=OJaRSw_N-7E
- 2- Comment faire une régulation par le logiciel STEP7 ? (ex. régulation de pression par la commande de mode de glissement) <https://www.youtube.com/watch?v=i1YHibESaKc>

PARTIE D : Etude comparative :

Le retour d'état plus l'action intégrale comme le régulateur PI sont deux commandes linéaires qui assurent la stabilité du système, et la poursuite de la référence grâce à l'action intégrale. Ces deux commandes ne sont pas si robuste que la commande par le mode de glissement, et en plus le retour d'état plus l'action intégrale manque de la rapidité lors de démarrage, par contre le régulateur PI, celui-là a un démarrage plus fort, et ça grâce à l'action proportionnelle qui donne une valeur importante à la commande quand l'erreur est grande.

La commande par mode de glissement est une commande robuste et qui converge en temps finis, mais elle a deux inconvénients : le premier est le phénomène de Chattering, celui-ci on peut l'éviter en remplaçant la fonction signe par l'arc-tangente, mais le deuxième est plus délicat, qu'on ne peut pas l'éviter c'est le fait que cette commande est énergétique plus que toutes les autres commandes qu'on a vue.

Pour la commande adaptative les erreurs de modélisation et les variations paramétrique, n'ont aucun effet sur la réponse du système « le système reste toujours stable » alors cette commande est dite robuste.

Recommandation pour le choix de la meilleure commande :

S'il s'agit d'un système linéaire stable, alors ça serai mieux d'éviter le mode de glissement qui est une commande énergétique, et le retour d'état plus l'action intégrale qui est une commande lente au démarrage, donc de préférence on utilise le PI qui est la commande la plus commode pour ce genre du système.

S'il s'agit d'un système linéaire dont les états interne risquent d'être en dehors de domaine de fonctionnement naturel, alors il est recommandé de maîtriser la dynamique de ces états, donc le retour d'état est la commande la plus commode.

Pour s'assurer que les deux commandes PI et RE+I assurent la robustesse contre les variations paramétrique et contre les erreurs de modélisation il suffit d'utiliser une commande adaptative indirecte, ou une adaptation par logique floue ou une synthèse par le loop-shaping pour le PI.

S'il s'agit d'un système non linéaire ou un système dont les erreurs de modélisation sont trop fortes, alors la commande de mode de glissement sera en mesure d'assurer la robustesse et les bonnes performances.

Conclusion :

Nous avons pu voir à travers ce chapitre l'application des commandes linéaire, non linéaire dans l'espace d'état et dans l'espace de transfert(Laplace) sur des différents systèmes réels à l'aide des outils de programmation offerts par STEP7 ainsi que la supervision des processus à travers l'interface graphique créé par le WinCC. Et finalement on a vu des recommandations sur le choix de commande à partir d'une étude comparative.

Conclusion générale

Nous avons vu dans ce mémoire les différentes étapes de régulations du système hydraulique CE117, équipé par un automate programmable S7 314 IFM qui nous a donné la possibilité de l'implémentation et la supervision de différentes commandes.

Notre travail était partagé en quatre chapitres :

Le premier chapitre avait pour but de présenter quelques généralités sur les automates programmables industriels.

Le deuxième chapitre avait pour but de présenter les différentes fonctionnalités des deux logiciels qu'on a utilisé lors de notre travail le Step7 et le WinCC.

Le troisième chapitre nous a permis en premier temps d'avoir une bonne connaissance sur les différents systèmes qui existe sur la station CE117 : les contraintes pour chaque système, les caractéristiques des capteurs, les différentes possibilités de réglage et finalement et après l'identification on a pu connaitre le modèle approprié pour chaque système, ce qui nous a aidés par la suite à la synthèse des régulateurs.

Dans le quatrième et le dernier chapitre nous avons pu voir l'application des commandes linéaire, non linéaire dans l'espace d'état et également dans l'espace de transfert (Laplace) sur des différents systèmes réels à l'aide des outils de programmation offerts par STEP7. Nous avons pu voir aussi la supervision des processus à travers l'interface graphique créé sous le WinCC. Et finalement on a vu des recommandations sur le choix de commande à travers une étude comparative.

Ce projet comporte de nombreux volets et touche à plusieurs disciplines en même temps, c'était une occasion formidable pour nous d'appliquer nos connaissances acquises le long de notre formation sur des systèmes réels qui ressemble à ceux qui sont disponible dans l'industrie, et de confronter à des problèmes qu'un ingénieur affronte dans le monde industriel.

En fin, nous espérons que notre travail sera utile aux étudiants en automatique, pour appliquer ce qu'ils prennent aux cours de l'informatique industrielle et d'Automatique avancé et pourquoi pas d'identification, et qu'il participe à forger des ingénieurs automaticien avec beaucoup d'expérience qui leurs permettre de transiter vers une nouvelle phase où ils seront obligés à manipuler des systèmes réels.

A la lumière des observations et des résultats présentes dans ce mémoire des perspectives intéressantes pouvant contribuer à l'amélioration du fonctionnement de notre dispositif expérimental, du point de vue de la commande :

- d'autres commandes peuvent être appliquées on cite le régulateur flou, la commande H_{∞} .

Du point de vue matériel :

- changer l'automate S7314 IFM par un autre automate qui possède plus de sorties analogiques, ou on lui rajoute un autre automate qui va être communiqué avec lui pour la commande de la station.
- Remplacer les vannes manuelles de la cuve agitée par des vannes proportionnelles
- Ajouter un clapet anti-retour à la sortie de la pompe.

Du point de vue amélioration :

- Programmer le mode de glissement tout en gardant le phénomène de Chattering, pour que les étudiants puissent saisir ce phénomène.
- Programmer une fenêtre pour l'identification des différents systèmes par différentes méthodes de la station sous le WinCC.
- Faire un archivage pour pouvoir identifier les différents systèmes à partir du WinCC.

Bibliographie

- [1] Niedercorn LT « la Briquerie » 57100 THIONVILLE-Automates programmables-présentation.
- [2] Slim BEN SAOUD : LES AUTOMATES PROGRAMMABLES INDUSTRIELS (API).
- [3] SIEMENS, Notions de base sur les bus de terrain avec SIMATIC S7-300, Annexe IV, Edition 05/2004
- [4] SIEMENS SIMATIC : Programmer avec STEP 7 Manuel
- [5] WinCC flexible 2008 SP2, WinCC flexible Information System.
- [6] www.siemens.com
- [7] : Automatique Contrôle et Régulation / Patrick Prouvost
- [8] FAHED ESHBAIR ‘ ‘ MODÉLISATION ET COMMANDE D'UN SYSTÈME MULTI-MOTEUR PAR LA TECHNIQUE DE COMMANDE *BACKSTEPPING*’ ’ EDITION AOÛT 2005
- [9] M.A.A.FOUKA, B.FERHAOUI, «REALISATION D'UNE MAQUETTE DE STATION DE POMPAGE A BASE D'AUTOMATE PROGRAMMABLE SIEMEN », projet de fin d'étude, Ecole Nationale Polytechnique 2015.
- [10] B.A.SAHAR, A.SOUCHANE «Commande et supervision de la station FESTO PCS_COMPACT à l'aide de STEP7 et WinCC», projet de fin d'étude, Ecole Nationale Polytechnique 2009.
- [11] M.BOUALBANI «Commande et supervision de la station CE117 Process Trainer avec l'automate s7 300», projet de fin d'étude, Ecole Nationale Polytechnique 2015.
- [12] A.ABRICHE, « Réalisation et gestion d'un prototype de station de pompage à base d'automates programmables industriels SIEMENS », projet de fin d'étude, Ecole Nationale Polytechnique 2007.

Manuels:

[13] Siemens, « STEP 7 Pour une transition facile de S5 à S7 » SIMATIC, Edition 12/2002

[14] Siemens, « Automate programmable S7-200 Manuel système » SIMATIC, Edition 08/2008.

[15] Siemens, « Mise en route avec le S7-1200 » SIMATIC, Edition 11/2009.

[16] Siemens, « SIMATIC S7-1500 une nouvelle génération automates » SIMATIC, Edition 2012.

[17] Siemens, « Systèmes intégrés compacts C7-623, C7-624 » SIMATIC, Edition 1995.

[18] Siemens, « Calculateur industriel M7-300 Installation et configuration –Caractéristiques des CPU » SIMATIC Edition 1997.

[19] Siemens, « Faites connaissance avec le S7-300 » SIMATIC, Edition 1996.

[20] Siemens, « S7-PLCSIM » version 5.4, SIMATIC, 07/2011.

[21] Siemens, « Mise en route WinCC flexible débutants » SIMATIC HMI, Edition 2008

[22] Siemens, « SIEMENS/fast industrie ; Automation and Drives-SCE », Edition 02/2006.

[23] Siemens, « Programmer avec STEP 7 » version 5.2, SIMATIC, 05/2010.

78

[24] Siemens, « STEP 7, getting started », version 5.1, SIMATIC, Edition 08/2000.

[25] Siemens, « Programmation d'automate avec SIMATIC S7-300-Notion de base » Edition : 03/2001.

[26] Siemens, « Programmation pas à pas du S7-GRAPH » Edition : 02/2002.

[27] Siemens, « SIMATIC WinCC, supervision de process avec Plant Intelligence » Brochure .Avril 2012

ANNEXE A

Les réseaux locaux industriels (MPI)

A. les réseaux locaux industriels (MPI)

Pour des installations complexes à nombre important de signaux d'entrées/sorties, il n'est aujourd'hui plus possible de réaliser des tâches d'automatisation avec une commande centrale unique.

On doit donc passer à une gestion des tâches de commandes réparties sur plusieurs automates plus petits. Ceux-ci seront ensuite coordonnés par des commandes de plus haut niveau ou par des calculateurs-mâîtres ; ils seront reliés à l'ensemble du procédé par un système à bus.

A.1.L'interface multipoint (MPI)

L'interface multipoint MPI (Multipoint Interface) est une des interfaces de communication intégrée SIMATIC S7 dans de nombreux appareils, connectés simultanément à plusieurs outils de programmation/PC avec STEP 7, les systèmes HMI (Operator Panel/Operator Station), S7-300, M7-300, S7-400 et M7-400.

Les domaines d'application du MPI et de PROFIBUS se recoupent, le MPI restant sensiblement meilleur marché, car cette interface est déjà disponible dans tous les produits SIMATIC S7. L'inconvénient notable du PROFIBUS est le fait que le protocole de transmission est un standard purement SIEMENS et que donc aucun produit de tout autre fabricant ne peut être intégré dans un tel système de bus.

L'entreprise SIEMENS fournit les caractéristiques suivantes pour la MPI :

- 32 participants MPI max.
- Procédé d'accès : Jeton (Token)
- Média de transfert : Câble blindé à deux fils (RS485), ou fibre optique (LWL / verre ou plastique)
- Vitesses de transmission de 19,2 Kbit/s par 187,5 Kbit/s jusqu'à 12Mbit/s
- Topologie du réseau : RS485 à bus ou à structure arborescente (avec des répéteurs), en utilisant une structure arbre LWL, étoile ou anneau
- Extension du réseau : Taille du segment de 50 m max., avec des répéteurs RS485 jusqu'à 1100 m, avec LWL par OLM jusqu'à 100 km
- Le nombre de liaisons possibles dynamiques pour la communication de base avec SIMATIC S7/M7- 300/-400 et de liaisons de communication statiques pour la communication étendue

aux PG/PC, systèmes SIMATIC HMI et SIMATIC S7/M7-300 dépend du type des CPU mises en places. [3]

A.2. Configuration d'un réseau MPI

La configuration d'un réseau MPI a l'aspect suivant :

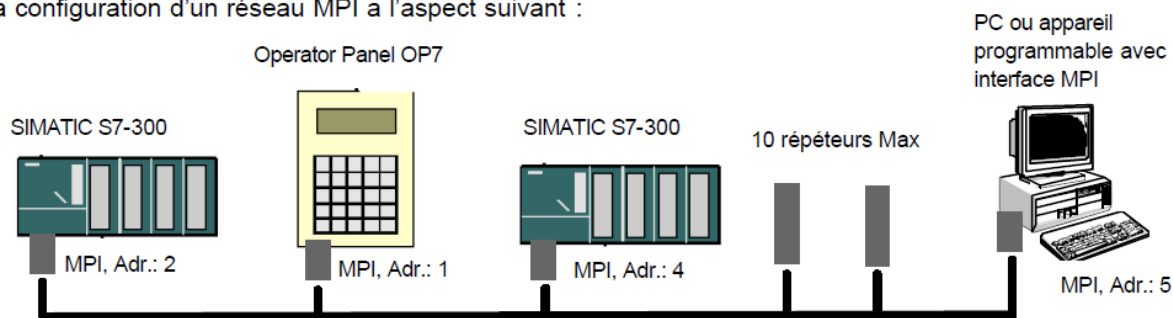


Figure A.1: Configuration d'un réseau MPI

- Jusqu'à 32 participants peuvent être connectés entre eux. Il faut toutefois garder à l'esprit que quelques processeurs de communication (CP) et modules de fonction possèdent également des adresses MPI dans le SIMATIC S7-300 et que donc le nombre de participant tient compte de celles-ci.
- Les adresses des différents participants peuvent être librement attribuées entre 0 et 31 (Configuration par défaut). [3]

A.3. Mise en service d'un SIMATIC S7-300 au réseau MPI

En ce qui concerne le SIMATIC S7-300, il faut veiller à ce que les processeurs de communication (CPs) et les modules de fonction (FMs) aient leurs propres adresses MPI. Celles-ci seront automatiquement attribuées et déterminées par la CPU, conformément à la série dans laquelle les modules sont classés dans les supports de modules, suivant le modèle suivant:

CPU : Adresse MPI

CP/FM1 : Adresse MPI + 1

CP/FM2 : Adresse MPI + 2

Ainsi, on obtient un accès direct de l'outil programmable aux modules connectés par la CPU. Cet accès s'effectue dans le S7-300 par un bus de communication interne (Bus K).

Une fois que les saisies du module matériel dans la **configuration matérielle** sont terminées, on peut procéder à l'accolage du SPS au réseau MPI. Pour cela, on doit suivre les étapes ci-dessous :

1. Le SPS doit être connecté à l'appareil de programmation par le MPI, mais ne doit pas encore se trouver dans le réseau MPI.
2. Double cliquez dans le tableau de configuration sur la CPU.
3. Cliquez ensuite sur ® Propriétés.
4. On va maintenant paramétrer l'adresse MPI de la CPU dans les propriétés de l'interface MPI (L'adresse MPI ne peut pas être plus élevée que l'adresse MPI configurée la plus haute !) et sélectionner le sous-réseau MPI.

Une fois les réglages de l'adresse MPI la plus élevée et du taux de baud du sous-réseau terminés, cliquez sur à Propriétés.

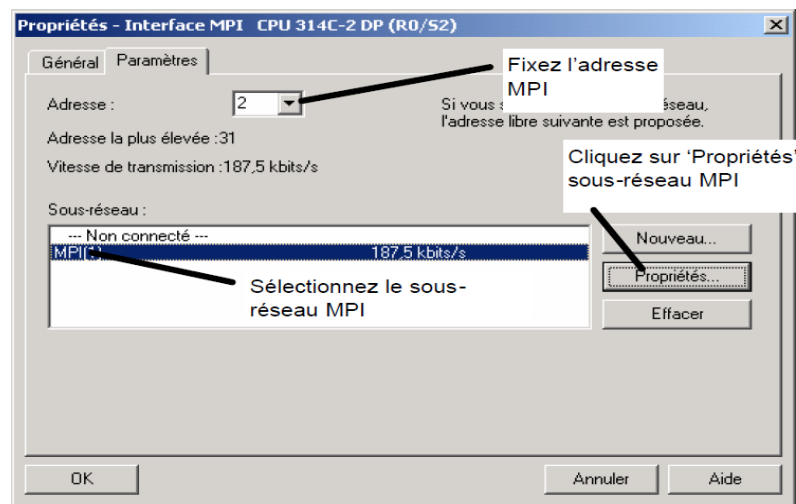


Figure A.2 : Fenêtre de propriété d'interface MPI

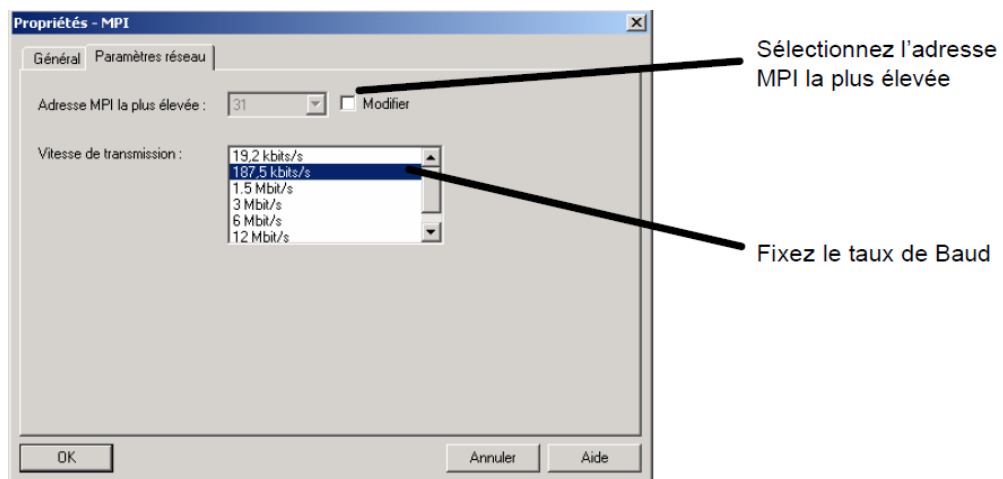


Figure A.3 : fenêtre de propriété MPI

A.4. Installation d'un appareil programmable / PC au réseau MPI

Il existe plusieurs possibilités pour connecter un appareil programmable des PC au réseau MPI. L'un d'entre eux est la carte d'enfichage MPI pour PCI ou PCMCIA ou le PC-Adapter alternatif qui peut être relié à une interface série.

Les appareils programmables PC doivent aussi être configurés pour être employés sur la MPI. On doit pour cela configurer les Adresses MPI, l'adresse MPI la plus élevée et le taux de Baud.

Dans l'exemple suivant, la configuration est réalisée pour un PC-Adapter :

- 1- Lancez « Paramétrage interface PG-PC » (→ Démarrer → SIMATIC → STEP7 → Paramétrage Interface PG-PC).
- 2- Sélectionnez le module qui est à disposition pour l'interface MPI (→ Sélectionner).
- 3- Choisissez le bloc souhaité par ex. 'PC-Adapter' et 'installer' (→ PC-Adapter → Installer).

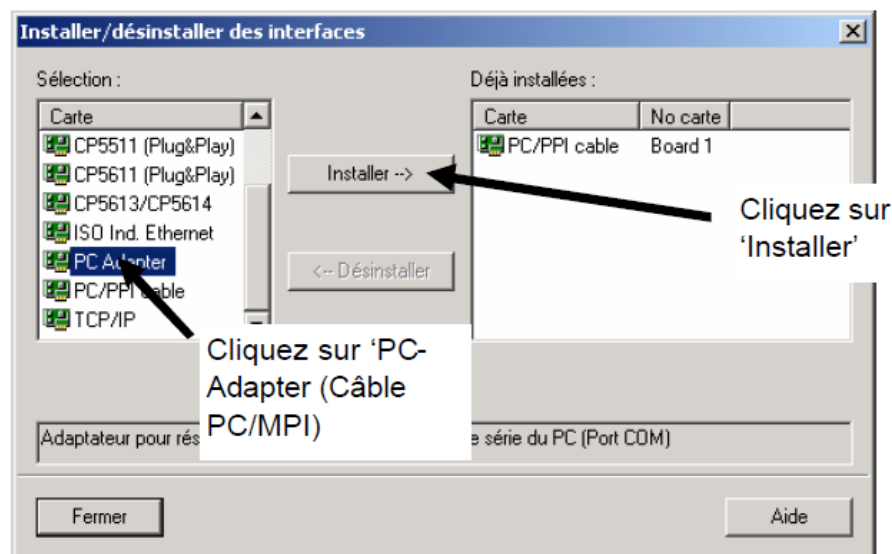


Figure A.4 : fenêtre de propriété MPI

ANNEXE B

**Diagnostic du processus par le logiciel
S7-PDIAG et diagnostic du matériel**

B.1.Diagnostic du processus par le logiciel S7-PDIAG

Dans S7-PDIAG, vous créez des surveillances vous permettant de détecter les erreurs survenant dans votre processus. Le diagnostic du processus fournit des informations sur le type, la localisation ainsi que sur la cause de l'erreur. Ces données de diagnostic constituent une aide précieuse pour l'élimination des erreurs. Voici les avantages qui en résultent :

- détection rapide et précise des erreurs du processus,
- réduction des temps de panne et des pertes de production dues à l'apparition d'erreurs,
- facilité de correction d'erreurs grâce à des informations ciblées,
- disponibilité accrue de l'installation.

Pour utiliser S7-PDIAG, procédez aux étapes suivantes :

1. Sélectionnez d'abord le type de surveillance approprié, puis créez une définition d'erreur dans laquelle vous décrivez exactement l'état d'erreur que vous souhaitez surveiller dans votre processus.

- Quel que soit le type de surveillance, sélectionnez d'abord l'opérande de début de diagnostic souhaité.

- Si vous optez pour une surveillance générale, programmez la logique de surveillance avec les éléments de langage S7-PDIAG.

- Si vous optez pour une surveillance du mouvement, complétez la logique de surveillance dans la boîte de dialogue affichée.

2. Définissez les opérandes d'arrêt et les opérandes à exclure, si vous en avez besoin.

3. Configurez ensuite les textes de message pour vos messages d'erreur.

4. Après avoir configuré toutes les définitions d'erreur avec les textes de messages correspondants, vous pouvez générer les blocs de surveillance contenant toutes les données requises par S7-PDIAG.

5. Insérez à présent l'appel des blocs de détection d'erreur à la fin de l'OB1 ou à l'endroit souhaité et chargez l'OB1 modifié ainsi que les blocs de surveillance générés par S7-PDIAG dans votre automate programmable.

6. Si l'état d'erreur se produit, un message d'erreur s'affiche sur tous les visuels (par exemple PG ou OP) avec le texte que vous avez défini.

7. Vous pouvez éventuellement modifier en ligne ou hors ligne les temps de surveillance paramétrés en utilisant la fonction "Modification".

B.1.1. Les modes de surveillance :

S7-PDIAG vous permet de détecter l'apparition d'erreurs précises dans le processus. Vous avez la possibilité de configurer ces erreurs pendant ou après la création de votre programme utilisateur. Vous disposez à cet effet de divers modes de surveillance :

B.1.1.a. La surveillance de l'opérande :

Vous surveillez des changements de niveau ou de front pour certains opérandes bien définis, surveillance que vous pouvez combiner avec un temps de retard. Vous pouvez surveiller des opérandes sans modifier votre programme utilisateur.

La surveillance de l'opérande est directement associée à un opérande appelé l'opérande de début de diagnostic.

La surveillance de l'opérande permet de vérifier si l'opérande de début de diagnostic a atteint un niveau donné après un temps donné (temps de surveillance). Si tel est le cas, l'erreur est signalée comme apparaissante. L'erreur disparaît lorsque l'opérande inverse de nouveau son niveau.

Selon que vous choisissez la surveillance du niveau ou du front, le temps de retard démarre immédiatement ou seulement après le front actif suivant.

Il existe deux types de surveillance de l'opérande :

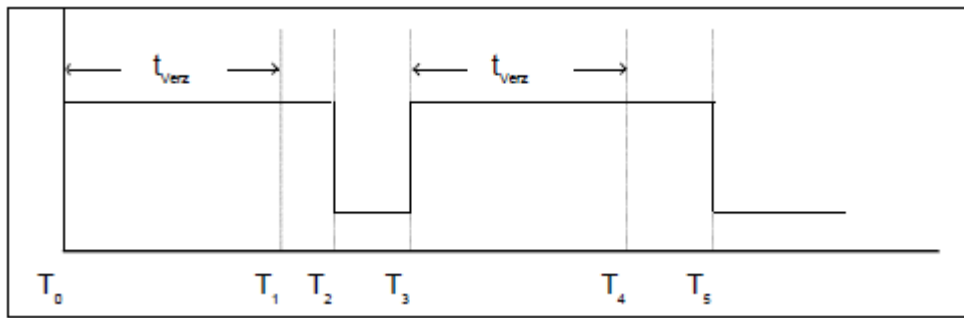
- la surveillance de niveau et
- la surveillance du front.

- **Surveillance de niveau :**

Dans la surveillance de niveau, c'est le niveau prédéfini (0 ou 1) de l'opérande de début de diagnostic qui est surveillé. L'état d'erreur survient lorsque l'opérande a pris le niveau indiqué durant un temps défini. Vous avez la possibilité de définir ce temps de retard. En cas de changement de niveau durant ce temps de retard, le temps est redémarré.

La surveillance commence avec le premier cycle (T_0 = Démarrage). Le temps de retard (t_{ret}) est respectivement démarré à T_0 et T_3 . Aussitôt que le niveau défini dure plus longtemps que le temps de retard défini, une erreur est détectée et signalée comme

"Apparaissante" (à T_1 et T_4). À T_2 et T_5 , l'erreur est signalée comme "disparaissante".



FigureB.1 : Le cycle de surveillance de l'opérande de niveau

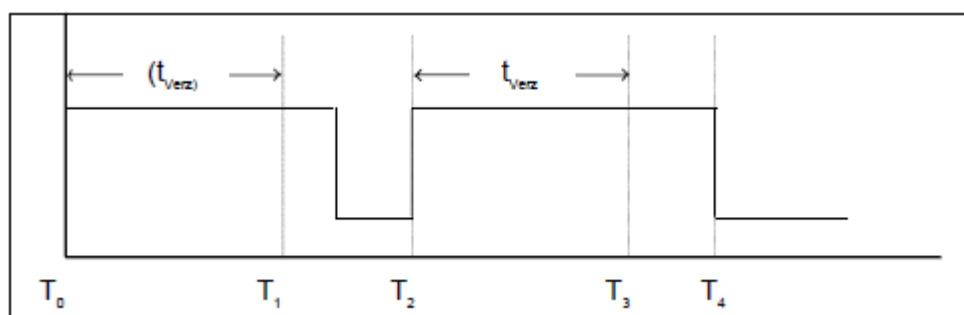
- **Surveillance du front :**

Dans la surveillance du front, c'est le front montant ou descendant défini (front "0 > 1" ou front "1 > 0") de l'opérande de début de diagnostic qui est surveillé. L'état d'erreur survient lorsqu'après un changement de front, l'opérande conserve le niveau erroné pendant le temps de retard donné (par exemple le niveau "1" après un front montant). En cas de changement de niveau durant ce temps de surveillance, le temps est redémarré.

La surveillance du front est par principe similaire à celle du niveau, si ce n'est que c'est l'état de l'opérande à surveiller qui est enregistré à l'instant T_0 (= Démarrage). Ceci signifie que le point de déclenchement n'est pas interprété comme front.

Le temps de retard (**tret**) ne démarre qu'après le front actif suivant (le front sélectionné).

Ainsi, une erreur n'est détectée qu'à T_3 et non pas déjà à T_1 et est ensuite signalée comme "apparaissante", comme cela serait le cas pour la surveillance du niveau. A T_4 l'erreur est signalée comme "disparaissante".



FigureB.2 : Le cycle de surveillance de l'opérande de front

B.1.1.b. La surveillance du mouvement :

Vous vérifiez que les mouvements physiques s'exécutent suffisamment rapidement et exactement. Le concept de surveillance du mouvement implique l'application de règles de programmation et requiert par conséquent l'adaptation de votre programme utilisateur.

S7-PDIAG contient des surveillances prédéfinies, spécialement prévues pour la surveillance du mouvement dans votre processus. Elles supposent que vous utilisiez les réseaux CONT pour la surveillance du mouvement ainsi que l'UDT_Motion.

La logique des surveillances du mouvement est préprogrammée et vous n'avez plus qu'à la compléter. Pour définir la surveillance du mouvement, vous reprenez une logique de surveillance prédéfinie que vous devez compléter et pouvez modifier. La situation d'erreur se produit lorsque les conditions définies sont remplies.

Vous pouvez choisir parmi les surveillances du mouvement prédéfinies suivantes :

- S7-PDIAG : surveillances de l'action surveille qu'un mouvement est terminé durant un temps d'action défini. Ceci est le cas lorsque la position finale cible est atteinte.
- S7-PDIAG : surveillances de la mise en route surveille qu'un mouvement débute effectivement lorsque toutes les conditions nécessaires sont remplies. Ceci est le cas lorsque la position finale actuelle est quittée durant le temps de mise en route prédéfini.
- S7-PDIAG : surveillances de la réaction surveille qu'une position finale cible atteinte reste stable sans qu'un amorçage n'ai lieu dans l'autre sens ou sans qu'elle ne soit quittée pendant une durée supérieure au temps de réaction prédéfini.
- S7-PDIAG : surveillances du verrouillage surveille que les conditions de verrouillage nécessaires au mouvement sont remplies.

Il est recommandé d'affecter un message à votre surveillance du mouvement. Ce message s'affichera sur le visuel après apparition de l'erreur.

1. Surveillance de l'action :

Il s'agit de l'une des quatre surveillances du mouvement. La surveillance de l'action surveille que la position finale à atteindre (Position finale cible) soit atteinte durant un temps défini (Temps d'action) après une commande de la machine (Déclencheur).

- Il s'agit de surveiller la position finale à atteindre. C'est donc le déroulement complet d'un mouvement du processus que l'on surveille.

- La logique de surveillance est prédéfinie. Il vous suffit de compléter le déclencheur et le temps d'action.

- La situation d'erreur se produit si le déclencheur était actif durant le temps d'action donné et si la position finale cible (début du diagnostic) n'a pas été activée.

Pour la surveillance de l'action, l'ODD est la <position finale>.

- **La logique de la surveillance de l'action est définie comme suit :**

ONDT (<déclencheur>, <temps d'action>)

AND

NOT <position finale cible>

L'opérande de début de diagnostic pour la position finale est "Final_Position[n]".

Dans ce cas, la logique de surveillance est prédéfinie comme suit :

ONDT (Nom_mouvement.Control1/2, <Temps d'action>)

AND

NOT Nom_mouvement.Final_Position[n]

"Nom_mouvement" est le nom de l'UDT_Motion et "Control1/2"

2. Surveillance de la mise en route :

Il s'agit de l'une des quatre surveillances du mouvement. La surveillance de la mise en route surveille que la position finale actuelle (Position finale effective) soit quittée durant un temps défini (Temps de mise en route) après une commande de la machine (Déclencheur).

- Il s'agit de surveiller la position finale actuelle. C'est donc la réaction d'un mouvement à une commande que l'on surveille. Cela signifie que l'erreur peut, le cas échéant, être détectée beaucoup plus tôt, contrairement à la surveillance de l'action. Ceci est tout particulièrement utile pour les processus lents.

- La logique de surveillance est prédéfinie. Il vous suffit de compléter le déclencheur et le temps de mise en route.

- La situation d'erreur se produit si le déclencheur était actif durant le temps d'action donné et si la position finale effective (début du diagnostic) est toujours active.

Pour la surveillance de la mise en route, l'ODD est la <position finale effective>.

- **La logique de surveillance de la mise en route est définie de la manière suivante :**

ONDT (<déclencheur>, <temps d'action>)

AND

<Position finale effective>

Si vous profitez des possibilités de programmation du mouvement offertes par S7-PDIAG et avez utilisé l'UDT_Motion dans votre programme, l'opérande de début de diagnostic pour la surveillance de la mise en route est la position finale "Final_Position[n]".

Dans ce cas, la logique de surveillance est prédéfinie comme suit :

ONDT (Nom_mouvement.Control1/2, <temps d'action>)

AND

Final_Position[n]

"Nom_mouvement" est le nom de l'UDT_Motion, "Control1/2" est le nom de la variable UDT

Control 1 ou Control 2 servant de déclencheur. Vous devez encore saisir le temps de surveillance souhaité.

3. Surveillance de la réaction

La surveillance de la réaction est l'une des quatre surveillances du mouvement. La surveillance de la réaction permet de vérifier si une position finale atteinte demeure stable après écoulement d'un temps donné (temps de réaction).

- Il s'agit de surveiller la position finale cible. La surveillance de la position finale cible requiert un memento de position supplémentaire précédant immédiatement la position finale effective dans le processus. C'est donc la réaction d'un mouvement à une commande que l'on surveille.

- La logique de surveillance est prédéfinie. Il vous suffit de compléter le memento de position et le temps de réaction.

- La situation d'erreur se produit lorsque, le memento de position étant activé, la position finale cible (début de diagnostic) ne peut être atteinte dans le temps de réaction indiqué ou lorsque la position finale est quittée plus longtemps que le temps de réaction ne le prescrit.

Pour la surveillance de la réaction, l'ODD est la <position finale cible>.

- **La logique de la surveillance de la réaction est définie comme suit :**

ONDT (<memento de position> **AND NOT** <position finale cible>, <temps de réaction>)

Si vous profitez des possibilités de programmation du mouvement offertes par S7-PDIAG et avez utilisé l'UDT_Motion dans votre programme, l'opérande de début de diagnostic pour la surveillance de la réaction est la position finale "Final_Position[n]".

Dans ce cas, la logique de surveillance est prédéfinie comme suit :

ONDT (Nom_mouvement.Position_Flag[n] **AND NOT**

Nom_mouvement.Final_Position[n], <Temps de réaction>)

"Nom_mouvement" est le com de l'UDT_Motion. Dans ce cas, vous n'avez plus qu'à saisir le temps de surveillance souhaité.

4. Surveillance du verrouillage

Il s'agit de l'une des quatre surveillances du mouvement. Elle permet de surveiller si la condition de verrouillage (exécutabilité est remplie après l'amorçage du mouvement (déclencheur) et après écoulement d'un temps donné (temps de verrouillage).

- C'est la surveillance de l'exécutabilité qui est réalisée. Il s'agit de surveiller l'exécutabilité, c'est à dire si la condition de verrouillage est remplie après amorçage du mouvement et après écoulement du temps de verrouillage.

- La logique de surveillance est prédéfinie. Il vous suffit de compléter le déclencheur et le temps de verrouillage.

- La situation d'erreur se produit si le déclencheur était actif durant le temps de verrouillage donné et si la condition de verrouillage (début du diagnostic) est encore inactive.

Pour la surveillance du verrouillage, l'ODD est l'<exécutabilité>.

- **La logique de surveillance du verrouillage est définie de la manière suivante :**

ONDT (<déclencheur>, <temps de verrouillage>

AND

NOT<exécutabilité>

Si vous profitez des possibilités de programmation du mouvement offertes par S7-PDIAG et avez utilisé l'UDT_Motion dans votre programme, l'opérande de début de diagnostic pour la surveillance du verrouillage est l'exécutabilité "Executability1/2"

Dans ce cas, la logique de surveillance est prédéfinie comme suit :

ONDT (Nom_mouvement.Trigger1/2, <Temps de verrouillage>)

AND

NOT Nom_mouvement.Executability1/2

"Nom_mouvement" est le com de l'UDT_Motion. Dans ce cas, vous n'avez plus qu'à saisir le temps de surveillance souhaité.

B.1.1.c. la surveillance générale :

Vous surveillez les erreurs du processus résultant de la combinaison de plusieurs opérandes, sans modifier votre programme utilisateur. S7-PDIAG émet un message d'erreur uniquement lorsque la combinaison logique est réalisée. La surveillance générale offre la possibilité de créer une logique

de surveillance sur mesure avec des éléments de langage propres à S7-PDIAG et de réaliser des surveillances d'erreur même complexes.

1. Logique de surveillance :

La surveillance générale vous permet de saisir votre propre logique de surveillance sous forme d'une suite d'expressions logiques. A l'aide des éléments de langage disponibles, vous pouvez créer une logique de surveillance autorisant une surveillance d'erreur complexe. L'état d'erreur survient lorsque les conditions définies sont remplies (logique = TRUE).

L'opérande de début de diagnostic sert uniquement de point de départ à l'analyse de critères. Si cet opérande doit faire partie de la surveillance (déclencher l'erreur), vous devez l'indiquer de manière explicite.

2. Etat d'erreur :

Dans la surveillance générale, l'état d'erreur se produit lorsque les conditions que vous avez définies sont remplies.

Pour la détection d'une erreur par une logique de surveillance définie, on a de manière générale :

- Le résultat logique "0" signifie qu'aucune erreur n'est détectée.
- le résultat logique "1" signifie qu'une erreur est actuellement détectée,
- Lors du changement du résultat logique de "0" à "1", un message d'erreur arrivante est toujours créé.
- Lors du changement du résultat logique de "1" à "0", un message d'erreur disparaissante est toujours créé.

3. Exemple de surveillance générale :

Dans la figure suivante, il s'agit p. ex. de surveiller si toutes les 3 grilles de protection d'une presse sont fermées.

On a comme hypothèses :

- Grille de protection 1 : E 1.0 à l'état 0 = grille de protection ouverte.
- Grille de protection 2 : E 3.5 à l'état 0 = grille de protection ouverte.
- Grille de protection 3 : E 7.2 à l'état 0 = grille de protection ouverte.
- Amorçage E 5.0 à l'état 1 = descente de l'estampe de presse
- La logique de surveillance s'écrit comme suit :

E 5.0 AND NOT (E 1.0 AND E 3.5 AND E 7.2)

Résultat : l'état d'erreur survient lorsque l'une des grilles de protection est ouverte.

4. Affectation de messages :

Il est recommandé d'affecter un message à vos définitions d'erreur, afin qu'il s'affiche sur les visuels connectée lorsqu'une erreur survient, c'est-à-dire lorsque la logique de surveillance est remplie. Programmation de surveillances générales.

5. Eléments de langage de S7-PDIAG :

Grâce aux éléments de langage, vous pouvez saisir votre logique de surveillance spécifique pour la surveillance générale.

Pour la saisie de la logique de surveillance, il est important d'utiliser exclusivement les éléments de langage appartenant au jeu de langage de S7-PDIAG et de les disposer d'après leur syntaxe correcte.

D'éventuelles erreurs de syntaxe dans la logique de surveillance sont signalées lorsque vous tentez d'enregistrer celle-ci. Tous les caractères valides dans STEP 7 comme désignation d'opérandes ou de temporisations sont autorisés.

Les éléments de langage suivants font partie du jeu de langage de S7-PDIAG :

- AND
- ONDT
- EN
- EP
- NOT
- OR
- SRT
- XOR
- Séparateurs
- Parenthèses
- Opérandes
- Temporisations
- Affectations Set et Reset

B.1.2.Présentation des UDT dans S7-PDIAG :

Un UDT (User Data Type) est un type de données utilisateur qui peut être enregistré comme bloc. Vous avez ainsi la possibilité d'utiliser plusieurs fois un UDT créé une fois pour toutes :

- d'une part en tant que format de données "normal",
- d'autre part en tant que modèle de création de blocs de même structure de données.

Les structures de données du diagnostic du processus avec S7-PDIAG sont définies par les UDT. Dans S7-PDIAG, vous disposez des UDT suivants :

B.1.2.a. UDT_Unit :

L'UDT_Unit contient les informations requises pour que le visuel (HMI) puisse affecter un message d'anomalie à l'endroit erroné du programme. L'UDT_Unit contient des définitions pour :

- la détection d'erreur groupée et l'acquiescement d'erreur groupée
- 16 modes de fonctionnement, dont deux sont prédéfinis comme modes "manuel" et "automatique". Vous pouvez définir les 14 autres modes de fonctionnement selon vos besoins.

B.1.2.b. UDT_S_Unit:

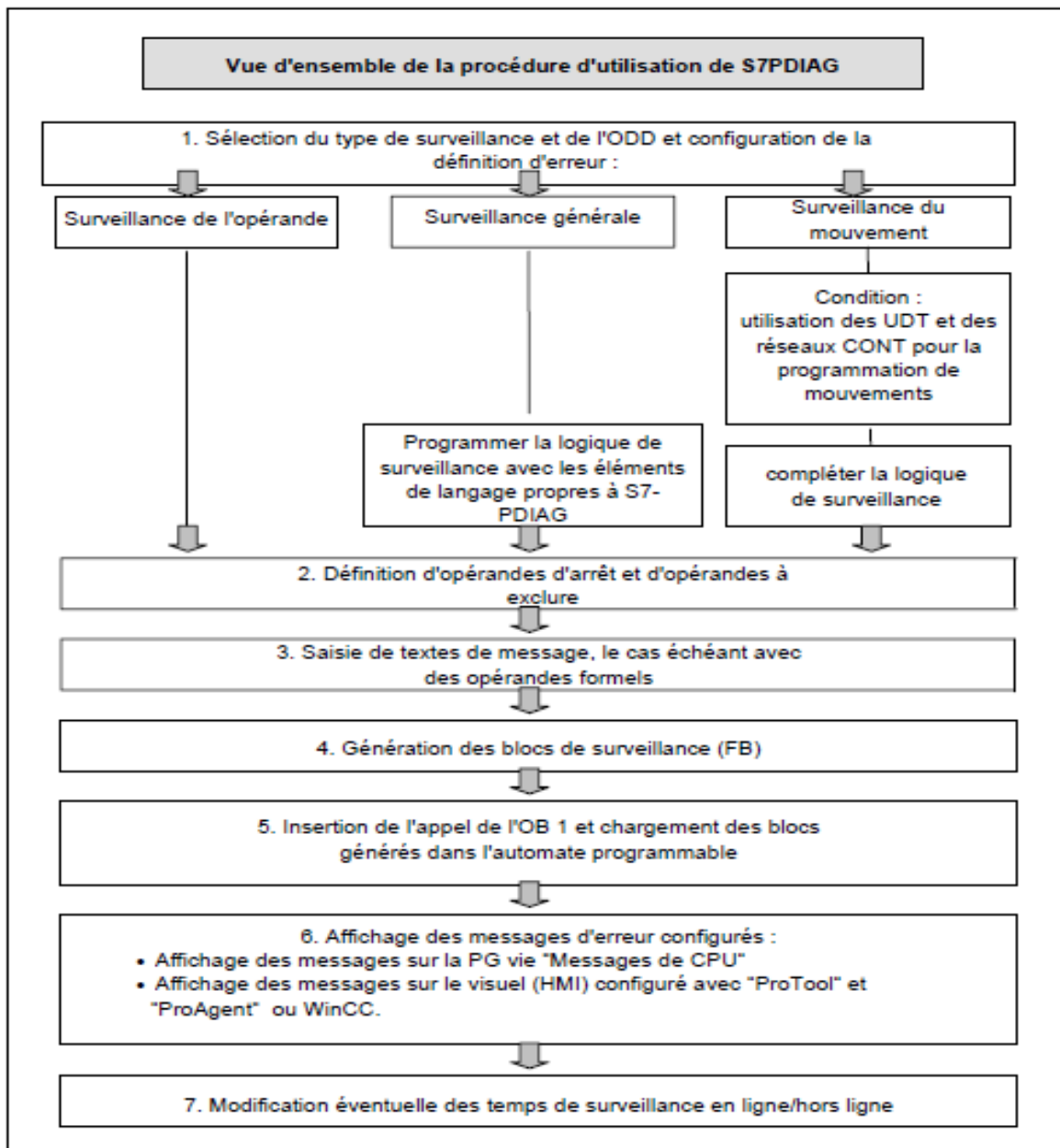
L'UDT_S_Unit contient l'adresse d'erreur groupée et l'acquiescement d'erreur groupée.

Ceci constitue un gain de mémoire et le mode de fonctionnement ne doit plus figurer dans toutes les unités partielles.

B.1.2.c. UDT_Motion :

L'UDT_Motion constitue une interface normée entre S7-PDIAG et les visuels (HMI) et possède les paramètres suivants pour :

- pouvoir représenter les mouvements dans des vues sur le visuel, (HMI) sans configuration supplémentaire et
- pouvoir traiter manuellement ces mouvements depuis les vues sur le visuel.

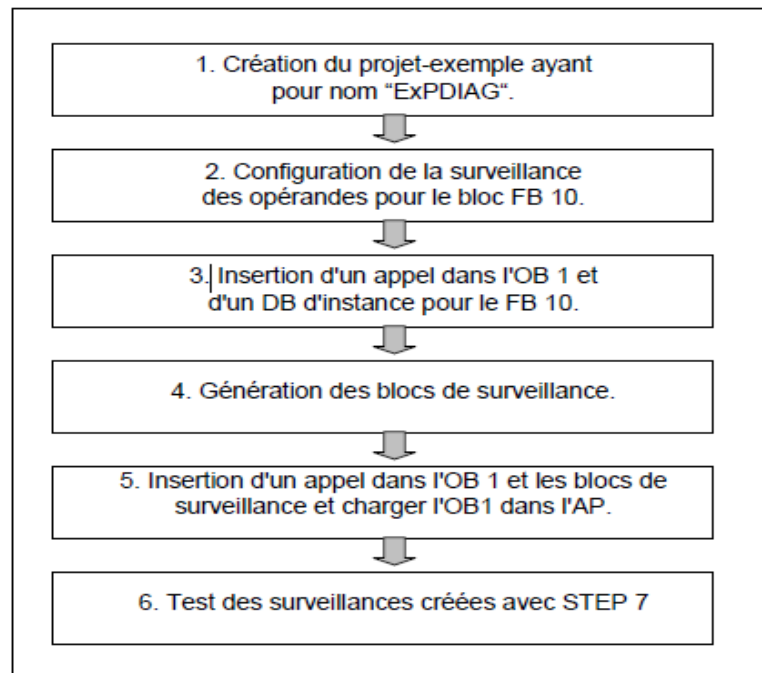


FigureB.3: Vue d'ensemble de la procédure d'utilisation de S7PDIAG

B.1.3. Exemple pour la mise en route de S7-PDIAG :

B.1.3. a. Présentation de la procédure

Les étapes suivantes sont à réaliser :



FigureB.4 : La procédure de configuration d'une surveillance de l'opérande avec S7-PDIAG

B.1.3. b. Création de l'exemple de projet et de programme

1. Création de l'exemple de projet

Créez d'abord un nouveau projet appelé "ExpPDIAG" dans SIMATIC Manager en vous servant de l'assistant STEP 7. Insérez un programme S7 sous votre configuration matérielle correspondante.

2. Création de l'exemple de programme S7

Dans SIMATIC Manager, sélectionnez le dossier Blocs de votre projet "ExpPDIAG" sous votre configuration matérielle et sous le programme S7, puis créez le bloc fonctionnel suivant en choisissant la commande **Insertion > Bloc S7 > Bloc fonctionnel** :

- FB 10

Il s'agit à présent de réaliser une surveillance de l'opérande dans le bloc précité. Pour que l'exemple soit exécutable sur l'automate programmable, l'octet d'entrée "0" et l'octet de sortie "1" doivent être connectés à des modules TOR. Si vous disposez uniquement d'une CPU, mais pas de modules TOR, insérez l'OB 122 (erreurs d'accès à la périphérie) et surveillez vos paramètres avec "Visualisation/forçage de variables".

- Programmation du FB 10 :

Ouvrez le FB 10 par double clic dans SIMATIC Manager, puis complétez la section des instructions dans l'éditeur "CONT/LIST/LOG" comme suit :

1. Entrez les commandes suivantes dans le premier réseau : nom de réseau : combinaison A1.0 dans le FB 10 programme : U E 0.0

U E 0.1

U E 0.2

U E 0.3

= A 1.0

Enregistrez le bloc en choisissant la commande Fichier > Enregistrer.

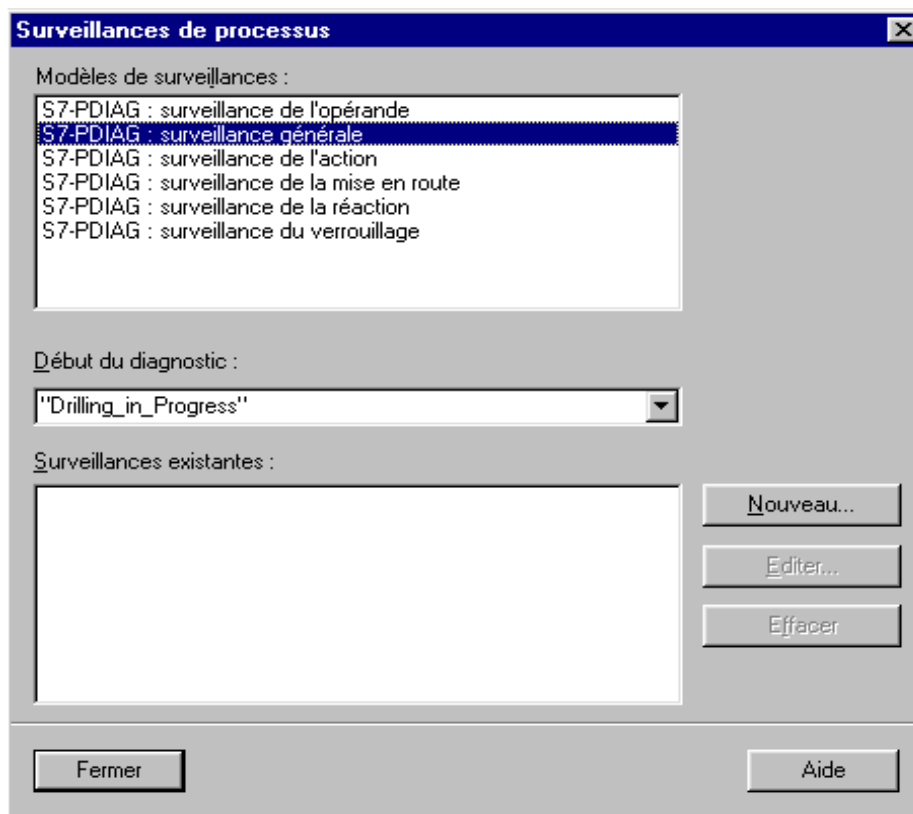
2.1. Configuration de la surveillance de l'opérande pour le FB10

Après avoir programmé le bloc pour l'exemple de programme, vous pouvez créer une surveillance de l'opérande pour ce bloc.

- **Procédure :**

1. Si le FB 10 n'est plus ouvert, ouvrez-le par double clic dans SIMATIC Manager. L'éditeur "CONT/LIST/LOG" s'ouvre.

2. Dans l'exemple, il s'agit de surveiller la sortie A 1.0. Nous allons donc insérer une surveillance de l'opérande pour cette sortie. Positionnez à cet effet le curseur dans la ligne d'affectation "= A 1.0" et appelez la boîte de dialogue "Surveillances de processus" en choisissant la commande **Edition > Propriétés spécifiques de l'objet > Surveillance**.

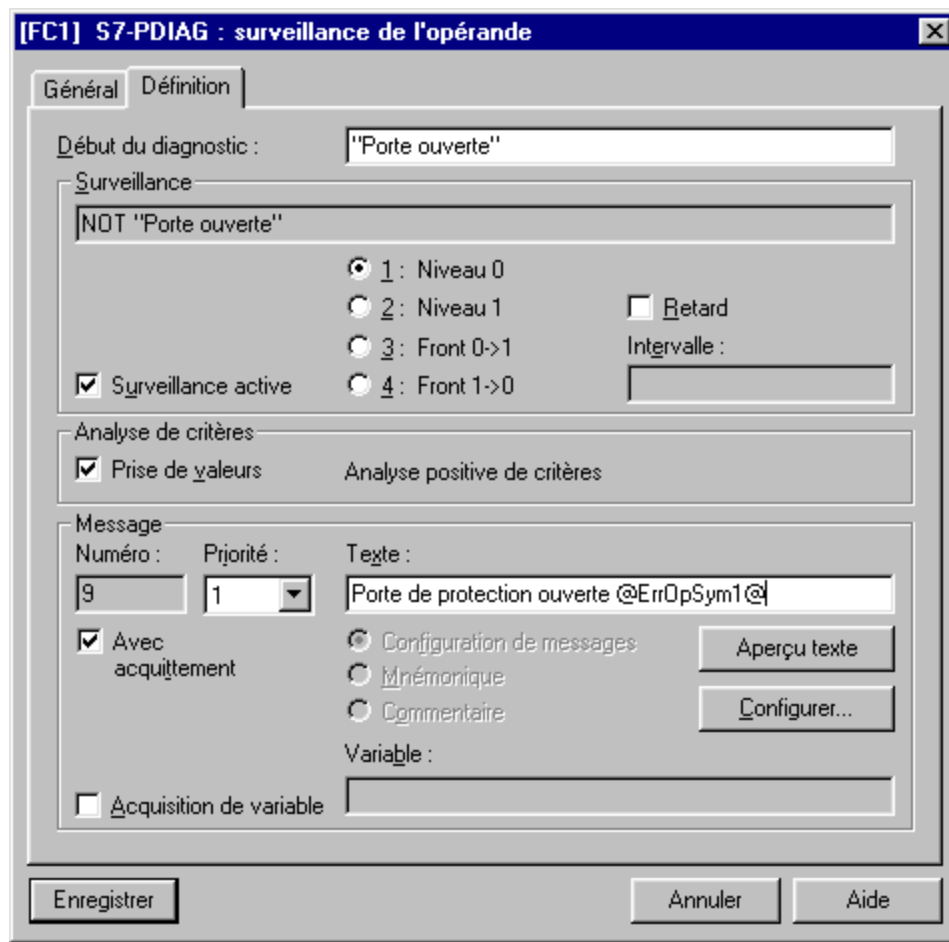


FigureB.5 : Vue de surveillances de processus

3. Dans la zone "Modèles", sélectionnez "S7-PDIAG : surveillance de l'opérande" et cliquez sur le bouton "Nouveau".

Résultat : la page d'onglet "Définition" s'affiche dans la fenêtre de dialogue "S7-PDIAG : surveillance de l'opérande". L'opérande de début de diagnostic indiqué dans la ligne d'affectation s'affiche, en l'occurrence "A1.0".

4. Pour affecter le texte de message correspondant à cette définition d'erreur, entrez "A 1.0 a le niveau 1 dans le FB 10" dans la zone du groupe "Message".



FigureB.6 : Fenêtre de la définition d'erreur

5. Quittez la page d'onglet OK. Vous venez de configurer une surveillance de l'opérande pour le niveau 1 de A 1.0. Ceci s'affiche à présent dans la boîte de dialogue "Surveillances du processus", sous "Surveillances existantes".
6. Quittez également la boîte de dialogue "Surveillances du processus" en cliquant sur "Fermer".
7. Enregistrez le bloc via la commande **Fichier > Enregistrer**, pour enregistrer la définition d'erreur venant d'être créée et quittez l'éditeur CONT/LIST/LOG.
8. Insérez à la fin de l'OB1 du projet "ExemplePDIAG" l'appel du FB10 suivant :
CALL FB 10, DB 10
9. Cliquez dans le dialogue suivant sur "Oui" pour créer le DB d'instance (ici le DB10) qui n'existe pas encore.

Résultat : Le DB10 a été créé avec les données spécifiques à S7-PDIAG et a également reçu l'attribut "S7_pdiag = true".

10. Enregistrez le bloc en choisissant la commande **Fichier > Enregistrer**, afin que la nouvelle définition d'erreur soit enregistrée dans le bloc, puis quittez l'éditeur CONT/LOG/LIST.

2.2. Génération des blocs de surveillance

Les étapes suivantes vous montrent comment créer des blocs de surveillance à partir de définitions d'erreur.

- **Procédure :**

1. Sélectionnez le dossier "Blocs" dans SIMATIC Manager et ouvrez S7-PDIAG en choisissant la commande **Outils > Configuration d'une surveillance de l'opérande**.

Résultat : les unités significatives pour PDIAG, en l'occurrence FB 10 et DB 10 s'affichent dans la hiérarchie des unités de S7-PDIAG.

2. Choisissez la commande **Diagnostic du processus > Compiler** dans S7-PDIAG. S'il s'agit de la première compilation, un message vous demande de vérifier les paramètres de compilation. Acquiescez ce message en cliquant sur "OK".

3. Dans la boîte de dialogue "Paramètres" qui s'ouvre et que vous pouvez également appeler en choisissant la commande **Outils > Paramètres**, entrez dans la page d'onglet "Modèles", le numéro "44" pour les blocs à compiler de la définition d'erreur et le numéro "45" pour les blocs de prise de valeur.

4. Quittez la boîte de dialogue par "OK". La progression de la compilation s'affiche et un message vous informe d'une éventuelle erreur.

Résultat : les blocs de surveillance créés et les SFC nécessaires s'affichent dans SIMATIC Manager.

B.2. Diagnostic du matériel et recherche d'erreurs : [4]

Des icônes de diagnostic vous permettent de déceler la présence d'informations de diagnostic pour un module. Elles indiquent l'état du module concerné et, pour les CPU, également leur état de fonctionnement.

Les icônes de diagnostic s'affichent dans la vue en ligne de la fenêtre du projet, dans la vue rapide (présélection) ou encore dans la vue de diagnostic lorsque vous appelez la fonction « Diagnostic du matériel ». Des informations de diagnostic détaillées sont données par l' « Etat du module » que vous appellerez par double clic sur une icône de diagnostic dans la vue rapide ou dans la vue de diagnostic.

a- Marche à suivre pour localiser les défauts :

1. Ouvrez la fenêtre en ligne du projet en choisissant la commande **Affichage > En ligne**.
2. Vérifiez pour quelle CPU une icône de diagnostic est affichée pour signaler une erreur ou un défaut. En appuyant sur la touche F1, vous obtenez une page d'aide avec les explications relatives aux icônes de diagnostic.
3. Sélectionnez la station que vous souhaitez examiner.
4. Choisissez la commande **Système cible > Diagnostic/Paramètres > Etat du module** pour afficher l'état du module de la CPU appartenant à cette station.
5. Choisissez la commande **Système cible > Diagnostic/Paramètres > Diagnostic du matériel** pour afficher la "vue rapide" avec la CPU et les modules défectueux de cette station. L'affichage de la vue rapide est présélectionné (commande **Outils > Paramètres**, page d'onglet "Affichage").
6. Sélectionnez un module défectueux dans la vue rapide.
7. Cliquez sur le bouton "Etat du module" pour obtenir les informations sur ce module.
8. Dans la vue rapide, cliquez sur le bouton "Station en ligne" pour afficher la vue de diagnostic. La vue de diagnostic affiche tous les modules de la station dans la disposition des emplacements.
9. Effectuez un double clic sur un module dans la vue de diagnostic pour en afficher l'état correspondant. Vous obtenez ainsi également des informations sur les modules non défectueux, qui ne sont donc pas affichés dans la vue rapide.

Il n'est pas impératif de réaliser la totalité de ces étapes et vous pouvez vous arrêter dès que vous avez trouvé l'information de diagnostic recherchée.

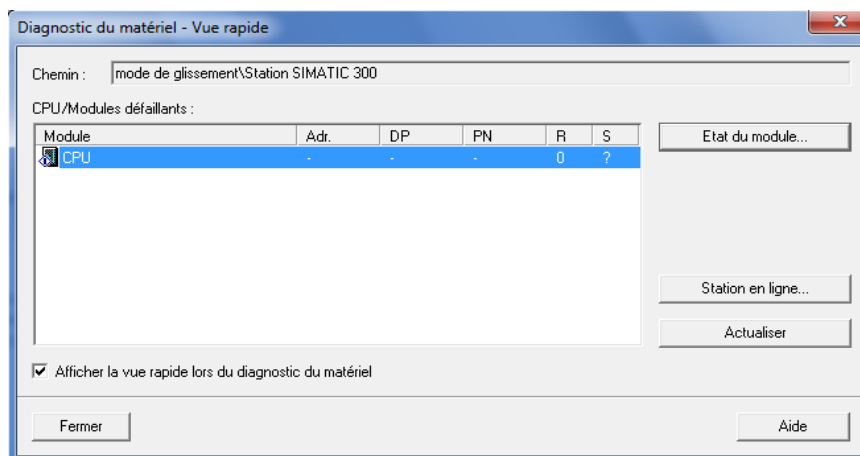
b- Actualisation de l'affichage des icônes de diagnostic :

La fenêtre correspondante doit être activée.

- Appuyez sur la touche de fonction F5 ou
- Choisissez la commande **Affichage > Actualiser** dans la fenêtre.

c- Appel de la vue rapide :

La vue rapide vous permet de parvenir rapidement dans le "Diagnostic du matériel" en fournissant des informations réduites par rapport aux informations complètes affichées dans HW Config. La vue rapide s'affiche par défaut à l'appel de la fonction "Diagnostic du matériel".



FigureB.7: vue rapide

d- Information de la vue rapide :

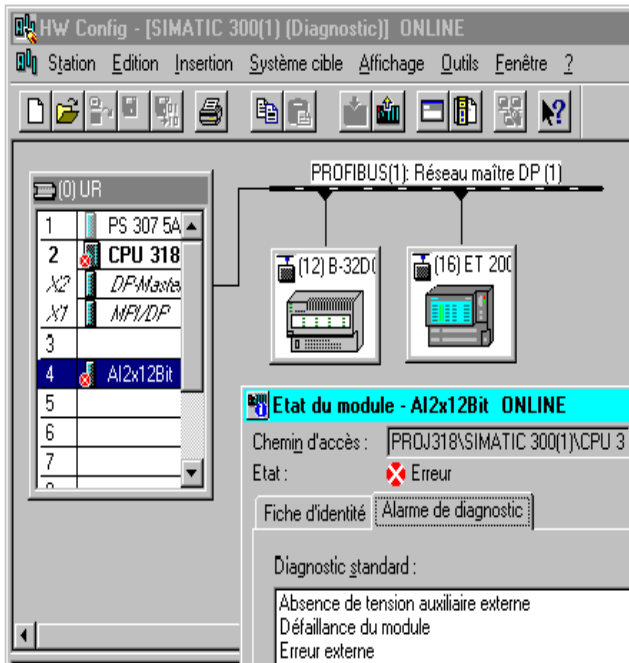
La vue rapide affiche les informations suivantes :

- Données pour la liaison en ligne à la CPU,
- Icône de diagnostic de la CPU,
- Icônes de diagnostic des modules pour lesquels la CPU a détecté un défaut (par exemple, alarme de diagnostic, erreur d'accès à la périphérie),
- type et adresse du module (profilé support/châssis, emplacement d'en fichage, réseau maître DP avec numéro de station).

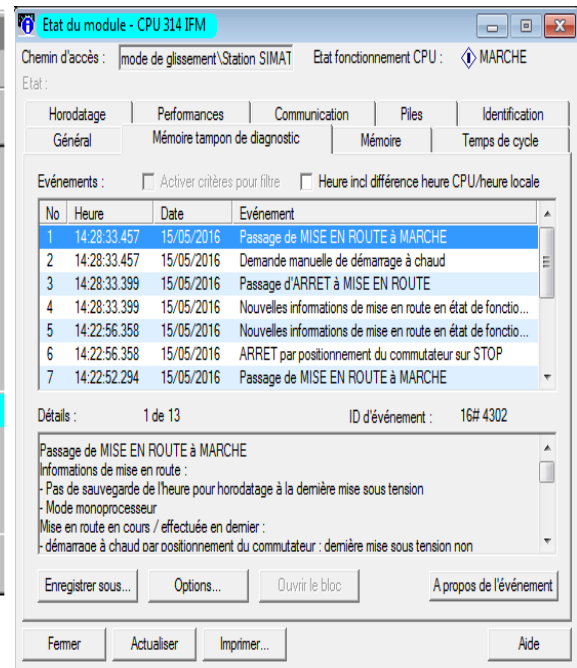
e- Autres possibilités de diagnostic dans la vue rapide :

• Affichage de l'état du module

Vous appelez cette boîte de dialogue en cliquant sur le bouton "Etat du module". Selon l'aptitude au diagnostic du module, vous y obtenez des informations détaillées sur le module sélectionné. L'état du module de la CPU vous permet en particulier d'afficher les entrées dans la mémoire tampon de diagnostic.



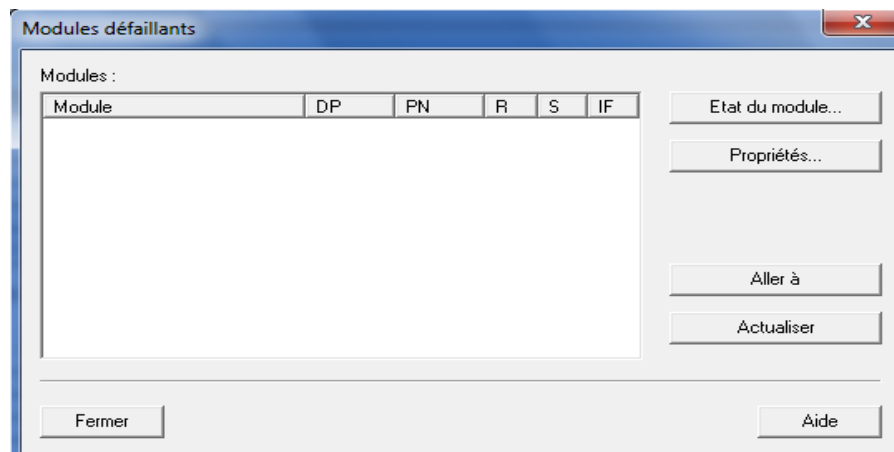
FigureB.8: fenêtre d'Etat du module lors d'une erreur



FigureB.9: fenêtre d'Etat du module

• Affichage de la vue de diagnostic

En cliquant sur le bouton "Station en ligne" vous appelez cette boîte de dialogue qui, contrairement à la vue rapide, fournit une représentation graphique de l'ensemble de la station ainsi que des informations sur la configuration. Vous êtes positionné sur le module qui est sélectionné dans la liste "CPU / Modules défaillants".



FigureB.10: fenêtre des modules défaillants