

Post-processeur ABB

Pour AUTOMGEN V6 et automates ABB CS31 KR31, KR91, et AC31 séries 40 et 50



CHAPITRE N°1.	3
INTRODUCTION	3
AMELIORATIONS ENTRE LES VERSIONS 6.20 ET 6.30	3
CHAPITRE N°2.	4
INSTALLATION	4
PARAMETRAGE	4
CONNEXION ENTRE LE PC ET LES AUTOMATES ABB	6
CHAPITRE N°3.	7
PRISE EN MAIN	7
CHAPITRE N°4.	8
GENERALITES	8
PREMIERE PHASE DE TRADUCTION DU GENERATEUR DE CODE	8
DEUXIEME PHASE DE TRADUCTION DU GENERATEUR DE CODE	8
MODULE DE DIALOGUE	8
EMULATEUR DE TERMINAL	9
CHAPITRE N°5.	10
LES QUATRE FICHIERS TEXTES ASSOCIES A UNE APPLICATION	10
LA DECLARATION UNITAIRE	10
LA DECLARATION DE TABLE LINEAIRE	11
LA DECLARATION DE TABLE POUR AFFECTATION AUTOMATIQUE	11
LES TYPES DE VARIABLES AUTOMGEN	11
LE FICHIER MODELE « DEFABB.VUS »	12
LES FICHIERS « .SRT » ET « .END »	13
LE FICHIER MODELE « DEFABB.SRT »	13
CHAPITRE N°6.	14
SYNTAXE DU LANGAGE ABB INTERMEDIAIRE	14
CHAPITRE N°7.	15
PARAMETRAGE DE L'AUTOMATE	15
DIRECTIVE DE SELECTION DU TYPE DE VARIABLE UTILISE PAR L'AUTOMATE	15
DIRECTIVE DE SELECTION DU TYPE D'AUTOMATE	15
CHAPITRE N°8.	16
COMPATIBILITE DES APPLICATIONS AVEC LE LOGICIEL DE PROGRAMMATION ABB	16

CHAPITRE N°9.	17
LES DIRECTIVES DE GENERATION DE CODE	17
;£E : GESTION DE L'ETAT DES ETAPES DANS LE TEMPS A LA CHARGE DU PROGRAMMEUR	17
CHAPITRE N°10.	18
LE MODULE DE DIALOGUE	18
REGLAGE DU LANGAGE UTILISE POUR DIALOGUER AVEC L'AUTOMATE ABB	18
DIAGNOSTIQUER LES ERREURS D'EXECUTION	18
CHAPITRE N°11.	20
TECHNIQUES AVANCEES	20
INSERTION DE MORCEAUX DE PROGRAMMES ECRITS EN LANGAGE ABB DANS UNE APPLICATION AUTOMGEN	20
REFERENCES AUX SYMBOLES DU FICHIER .SYM DANS LES SECTIONS ECRITES EN LANGAGE CONSTRUCTEUR	21
UTILISATION DES TACHES RAPIDES SUR LES AUTOMATES AC31	21
CHAPITRE N°12.	23
LIMITE DE COMPATIBILITE	23
CHAPITRE N°13.	24
UTILISATION DES E/S ANALOGIQUES SUR UNE EXTENTION DE L'AUTOMATE AC31	24
CHAPITRE N°14.	26
EXEMPLE COMPLET	26

CHAPITRE N°1.

Introduction

Le post-processeur ABB est un module logiciel composé de plusieurs fichiers exécutables.

Il traduit les applications développées avec l'atelier logiciel AUTOMGEN en code ABB et les transfère dans un automate cible.

Les automates ABB CS31 KR31 et KR91, ainsi que les automates AC31 série 40 et 50 sont utilisables avec le post-processeur.

Le post-processeur permet d'effectuer la mise au point des applications depuis l'atelier logiciel AUTOMGEN en donnant accès aux fonctions de visualisation dynamique, de changement d'état des variables, de forçage des variables et de modification du mode de marche.

Le post-processeur ABB est développé par IRAI en collaboration technique avec la société ABB.

La société IRAI assure exclusivement le support technique du logiciel AUTOMGEN.

Améliorations entre les versions 6.20 et 6.30

De multiples améliorations ont été apportées sur la version 6.30 du post-processeur ABB.

En effet cette nouvelle version supporte toujours les automates ABB CS31 KR31 et KR 91, mais le post processeur permet également d'utiliser les automates AC31 séries 40 et 50.

- L'utilisation des bits de mot et bits d'accumulateur est maintenant possible sur les automates CS31 et AC31.
- Les automates AC31 série 40 et 50 permettent l'utilisation des tâches rapides et des entrées sorties analogiques sur les extensions du type XM06B5.
- Pour le transfert du programme dans l'automate : la mise en EEPROM, obligatoire sur les automates AC31 est faite automatiquement, contrairement au automates CS31 où cela est toujours optionnel. Dans les deux cas une barre d'avancement est présente et la quantité de mémoire libre restante est affichée en fin de transfert.
- En mode connecté, la visualisation dynamique des temporisations permet l'affichage des valeurs courantes de celle-ci, sur les automates AC31 séries 40 et 50, tandis que pour les automates CS31, c'est la consigne qui est affichée.
- Les entrées sorties booléennes peuvent, maintenant, être forcées.

CHAPITRE N°2.

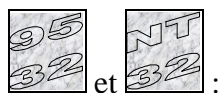
Installation

Pour installer le post-processeur ABB sur votre disque dur, procédez de la façon suivante :

1. Placez la disquette repérée « Post-processeur ABB » dans le lecteur de disquette A ou B,
2. Lancez l'exécution de « IRAINST. EXE »
3. Suivez ensuite les instructions du programme d'installation.

Paramétrage

Une fois l'installation terminée, lancez l'environnement AUTOMGEN suivez les instructions suivantes :



1. Sélectionnez la commande « Cibles ... » du menu « Option »,
2. Choisissez la ligne « ABB » puis cliquez sur « OK »,
3. Choisissez le menu « Options », « Paramètres par défaut », le bouton poussoir « ABBDIA32.EXE »,
4. La boîte de dialogue de paramétrage du module de dialogue s'ouvre .



1. Sélectionnez la commande « Cibles ... » du menu « Option »,
2. Cliquez sur le bouton « Insérer »,

La fenêtre suivante est alors affichée :

A screenshot of a Windows-style dialog box titled 'Définition d'une cible'. It contains three text input fields. The first field is labeled 'Intitulé du post-processeur'. The second field is labeled 'Nom du post-processeur, éventuellement du module de configuration'. The third field is labeled 'Nom de l'exécuteur, éventuellement de l'outil de maintenance'. At the bottom left is a button with a question mark '?'. At the bottom right are two buttons labeled 'Annuler' and 'OK'.

3. Entrez sur la première ligne l'intitulé du post-processeur. Cette ligne n'est pas fonctionnelle, elle permettra simplement de sélectionner le post-processeur dans l'environnement. Il est pratique d'entrer un nom générique (« **ABB** » par exemple) suivi du port de communication utilisé (« **COM2** » par exemple). Entrez par exemple « **ABB sur COM1** » ou « **ABB sur COM2** »,

4. Entrez sur la deuxième ligne le nom du générateur de code ABB. Entrez « **PPABB.EXE** »,
5. Entrez sur la troisième ligne le nom du module de communication suivi d'une virgule et du nom de l'émulateur de terminal. Entrez par exemple « **ABBDIAL.EXE,ABBTERM.EXE** » pour utiliser le port COM1 ou entrez « **ABBDIAL.EXE -C2,ABBTERM.EXE /C2** » pour utiliser le port COM2. Il faut un et un seul espace entre « **ABBDIAL.EXE** » et « **-C2** ». Il faut un et un seul espace entre « **ABBTERM.EXE** » et « **/C2** ».

Exemple d'installation du post-processeur ABB connecté sur COM1.

The screenshot shows a Windows-style dialog box titled "Définition d'une cible". It contains three text input fields. The first field, labeled "Intitulé du post-processeur", contains the text "ABB sur COM1". The second field, labeled "Nom du post-processeur, éventuellement du module de configuration", contains the text "PPABB.EXE". The third field, labeled "Nom de l'exécuteur, éventuellement de l'outil de maintenance", contains the text "ABBDIAL.EXE,ABBTERM.EXE". At the bottom left is a question mark icon in a box. At the bottom right are two buttons: "Annuler" and "OK".

Exemple d'installation sur COM2.:

The screenshot shows the same "Définition d'une cible" dialog box, but with different values. The first field "Intitulé du post-processeur" contains "ABB sur COM2". The second field "Nom du post-processeur, éventuellement du module de configuration" contains "PPABB.EXE". The third field "Nom de l'exécuteur, éventuellement de l'outil de maintenance" contains "ABBDIAL.EXE -C2,ABBTERM.EXE /C2". The layout, including the question mark icon and "Annuler"/"OK" buttons, is identical to the previous screenshot.

6. L'installation est terminée, cliquez sur le post-processeur que vous souhaitez utiliser, puis sur le bouton « OK ».

Si vous voulez vérifier ou modifier l'installation d'un post-processeur, cliquez sur la ligne correspondante, puis sur le bouton « Modifier ».

Connexion entre le PC et les automates ABB

Pour pouvoir connecter AUTOMGEN aux automates ABB, il faut utiliser un câble RS232.

Consultez la documentation ABB pour connaître le câblage en fonction du type d'automate utilisé.

Pour l'utilisation de la version 16 bits d'AUTOMGEN seuls les trois fils « Emission des données », « Réception des données » et « Masse » doivent être câblés côté PC. Le câblage des signaux de contrôle (RTS, CTS, etc ...) peut entraîner des problèmes de fonctionnement suivant la configuration de WINDOWS. Ce problème ne concerne pas la version 32 bits d'AUTOMGEN.

CHAPITRE N°3.

Prise en main

Ce chapitre donne le déroulement précis permettant de compiler un fichier d'exemple et de l'exploiter sur un automate ABB.

Si vous n'avez pas installé le programme d'exemple, relancez la procédure d'installation pour le faire.

1. Lancez l'environnement AUTOMGEN,
2. Cliquez sur l'option « Ouvrir Folio » du menu « Fichier »,
3. A l'aide du sélecteur de fichiers, ouvrez le folio « ABB.GR7 » qui se trouve dans le sous-répertoire « EXABB » du répertoire où est installé AUTOMGEN (probablement « C:\AUTOMV6 »¹),
4. Cliquez sur l'option « Cibles ... » du menu « Option », sélectionnez la ligne « ABB »² puis le bouton « OK »,
5. Cliquez sur l'option « Paramètres par défaut » du menu « Option », cliquez sur l'option « Post-processeur PPABB32.EXE ... », répondez « Oui » à la question posée et placez vous dans le fichier « DEFFABB.VSY ». Enlevez £ si vous utilisez un automate CS31. Derrière « \$CPU= », écrivez AC pour les automates AC31 ou CS pour les automates CS31 pour choisir les plages des variables.
6. Cliquez sur l'option « Compiler » du menu « Compiler »,

La compilation est lancée : elle se termine par l'appel du post-processeur ABB.

Si le post-processeur n'est pas lancé, c'est probablement que l'installation n'a pas été réalisée correctement, reprenez point par point le chapitre « Installation ».

Si tout s'est bien passé, un fichier « ABB.COD » a été généré dans le même répertoire que le fichier « ABB.GR7 ». C'est ce fichier qui sera relu et transféré dans l'automate.

7. Cliquez sur la commande « Charger » du menu « Exécuter »,
6. Une fenêtre « Module de dialogue ABB » s'ouvre et le transfert est réalisé³,
7. Cliquez sur la commande « Initialiser » du menu « Exécuter »,
8. Cliquez sur la commande « Exécuter » du menu « Exécuter », l'automate passe en RUN,
9. Cliquez sur la commande « Visualiser » du menu « Debug », la visualisation dynamique est active sur le folio « ABB.GR7 ».

Ceci termine le chapitre « Prise en Main ».

¹si le fichier n'existe pas, vous n'avez pas sélectionné l'option « Installer les exemples » dans la procédure d'installation. Relancez la procédure d'installation dans ce cas.

²si cette ligne n'apparaît pas, c'est que l'installation n'a pas été réalisée correctement, retournez au chapitre « Installation ».

³si ce n'est pas le cas, vérifiez que l'installation est en accord avec le port de communication utilisé et la configuration de l'automate (voir le chapitre « Installation »). Le langage de dialogue de la prise console de l'automate doit être « langue allemande » (voir le chapitre « module de dialogue » pour plus de détails).

CHAPITRE N°4.

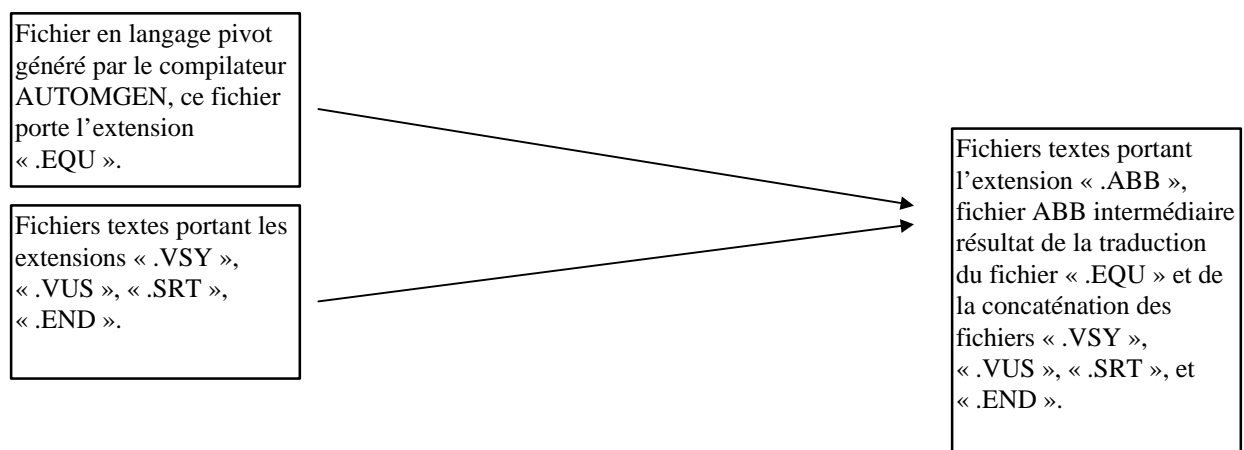
Généralités

Le post-processeur se compose de différents modules détaillés ci-dessous. Pour rendre les explications plus compréhensibles, seuls les fichiers principaux et accessibles au programmeur seront mentionnés.

Première phase de traduction du générateur de code

« PPABB.EXE » ou « PPABB32.EXE » sont les exécutables qui réalisent la première passe de traduction. Ils traduisent le langage pivot d'AUTOMGEN représenté par un fichier portant l'extension « .EQU » en fichier ABB intermédiaire qui porte l'extension « .ABB ». En outre, ce générateur de code cherche des fichiers textes qui portent les extensions « .VSY », « .VUS », « .SRT », « .END » et les recopie dans le fichier de sortie s'ils existent. La création de ces fichiers est à votre charge et sera détaillée dans le chapitre suivant.

Le schéma ci-dessous donne une description de la tâche accomplie par « PPABB.EXE » ou « PPABB32.EXE » :



Deuxième phase de traduction du générateur de code

« ABBCOMP.EXE » (16 bits) ou « ABBCO32.EXE » (32 bits) sont les exécutables qui réalisent la deuxième phase de traduction. Ils traduisent le fichier « .ABB » en un fichier « .COD ». Ces traducteurs effectuent deux tâches : remplacer le nom des variables AUTOMGEN par le nom des variables ABB et résoudre les sauts dans le programme. Un fichier « .LAP » est également généré, il contient la liste des instructions en langage ABB.

Module de dialogue

« WABBDIAL.EXE » (16 bits) ou « ABBDIA32.EXE » (32 bits) sont les modules de dialogue appelés par l'environnement AUTOMGEN pour dialoguer sur le port série du PC. Ces modules sont utilisés pour les fonctions de téléchargement, de visualisation dynamique, de changement d'état des variables et de forçage des modes de marche de l'automate.

Emulateur de terminal

« ABBTERM.EXE » est un émulateur de terminal qui permet de dialoguer avec l'automate en utilisant les commandes définies dans le protocole ABB (voir documentation ABB).

CHAPITRE N°5.

Les quatre fichiers textes associés à une application

Comme il a été dit au chapitre précédent, le générateur de code récupère quatre fichiers textes lors de la première phase de traduction.

Des fichiers standards, utilisables pour la grande majorité des applications, sont fournis.

Si vous êtes débutant dans l'utilisation d'AUTOMGEN ou du post-processeur ABB, nous vous conseillons d'utiliser ces fichiers sans les modifier. Dans un deuxième temps, et si vous désirez réaliser des applications particulières (optimisation de la taille du programme, tests de fronts sur les entrées sorties, ajout de morceaux de programmes en langage ABB), vous pourrez étudier ce chapitre ainsi que le chapitre « Techniques avancées ».

Le générateur de code cherche les fichiers portant l'extension « .VSY », « .VUS », « .SRT » et « .END ».

Pour chacun de ces quatre fichiers, le générateur de code recherche :

- d'abord un fichier portant le même nom que le premier folio de l'application,
- si ce fichier n'existe pas, c'est le fichier portant le nom « DEFABB » et qui se trouve dans le répertoire où a été installé AUTOMGEN qui est sélectionné,
- si ni l'un ni l'autre de ces fichiers existe, alors le générateur de code considère cette partie comme vide et ne génère aucun message d'erreur.

Ces fichiers peuvent recevoir des commentaires placés à droite d'un caractère « ; » (point virgule).

Les fichiers « .VSY » et « .VUS » sont des fichiers de déclaration de variables. Ils donnent la correspondance entre les variables AUTOMGEN et les variables ABB.

Le fichier « .VSY » donne la correspondance pour les variables dites « Système », le fichier « DEFABB.VSY » est fourni en modèle dans le répertoire où est installé AUTOMGEN.

Le fichier « .VUS » donne la correspondance pour les variables utilisées dans l'application. Le fichier « DEFABB.VUS » est fourni en modèle dans le répertoire où est installé AUTOMGEN.

Si une ou plusieurs variables utilisées dans l'application n'ont pas de correspondance, alors le générateur de code signalera des erreurs.

Il existe plusieurs syntaxes permettant de déclarer les correspondances de variables :

La déclaration unitaire

Elle donne la correspondance entre une variable AUTOMGEN et une variable Automate.

La syntaxe est :

variable AUTOMGEN = variable automate

Exemple :

#m200=MW0,0 ; le mot AUTOMGEN 200 est attribué au mot ABB MW0,0

#i10=E62,0 ; l'entrée 10 d'AUTOMGEN est attribuée à l'entrée ABB E62,00

La déclaration de table linéaire

Elle donne la correspondance entre une série de variables AUTOMGEN et une série de variables Automate.

La syntaxe est :

longueur de la table : première variable AUTOMGEN = première variable automate

Exemple :

##16:m100=MW1,0 ; les mots AUTOMGEN 100 à 115 sont attribués aux mots ABB
; MW1,0 à MW1,15

La déclaration de table pour affectation automatique

Elle laisse le soin au générateur de code d'affecter un ou plusieurs types de variables dans une table de variables Automate. Le générateur de code affecte automatiquement les variables suivant ses besoins et si la table n'est pas saturée.

La syntaxe est :

type(s) de variable AUTOMGEN = première variable ABB : dernier numéro de variable ABB

Un type est représenté par un nom de variable sans numéro. Si plusieurs types sont à préciser, alors ils doivent être séparés par le caractère « & ».

Exemples :

###x&bx=M0,4:21,15; affecte automatiquement les bits d'étapes d'AUTOMGEN dans les
; bits ABB M0,4 à M21,15

###m&c=MW0,2:5,15; affecte les mots et les compteurs d'AUTOMGEN dans les mots
; ABB MW0,2 à MW5,15

Ce type de déclaration est moins prioritaire que les types « # » et « ## ».

Plusieurs déclarations de ce type peuvent être utilisées pour un même type de variable AUTOMGEN. Dans ce cas, le générateur de code remplira dans l'ordre de déclaration et suivant ses besoins, une ou plusieurs tables.

Les types de variables AUTOMGEN

Les types de variables AUTOMGEN forment un sur-ensemble par rapport aux types décrits dans le manuel d'utilisation du module principal dans la partie « Compilateur ».

En voici la liste exhaustive :

X	étape, état actuel
Bx	étape, état futur
Dx	bit de test de front montant sur une étape
Ux	bit de test de front descendant sur une étape
b	bit, état actuel

bb	bit, état futur
u	bit utilisateur, état actuel
bu	bit utilisateur, état futur
i	entrée, état actuel
bi	entrée, état futur
o	sortie, état actuel
bo	sortie, état futur
m	mot
c	compteur
t	temporisation, état de fin
bt	temporisation, état de lancement
tconsi	temporisation, mot de consigne
ui	bit de test de front montant sur une entrée
di	bit de test de front descendant sur une entrée
uo	bit de test de front montant sur une sortie
do	bit de test de front descendant sur une sortie
ut	bit de test de front montant sur une temporisation
dt	bit de test de front descendant sur une temporisation

Pour les variables booléennes d'AUTOMGEN, deux variables booléennes ABB sont utilisées afin de gérer les états dans le temps. Pour les entrées et les sorties d'AUTOMGEN, il est possible d'affecter une même variable ABB dans le cas où aucun test de front n'est utilisé sur ces types de variables.

Le fichier modèle « DEFABB.VUS »

Détail du fichier « DEFABB.VUS » présent dans le répertoire où est installé AUTOMGEN.

; Fichier .VUS pour automate ABB

; bits (m0,0 r,serv, dans le .VSY)

###i&bo&x&bx&b&bb&u&bu&t&bt&ui&di&uo&do&ux&dx&uu&du&ub&db&tt&ut&dt=m0,1:21,15

###i&bo&x&bx&b&bb&u&bu&t&bt&ui&di&uo&do&ux&dx&uu&du&ub&db&tt&ut&dt=m230,0:239,15

; mots et compteurs (mw0,0 r,serv, dans le .VSY)

###c&m=mw0,1:5,15

###c&m=mw230,0:239,15

; doubles mots (dw0,0 r,serv, dans le .VSY)

\$IF CPU=CS

###l=md0,1:1,15

\$ENDIF

\$IF CPU=AC

###l&tcompt=md0,1:1,15

\$ENDIF

*Définition des variables
booléennes autres qu'entrées
sorties*

Définition des mots et compteurs

Définition des doubles mots

```
; 16 temporisations  
##16:tconsi0=kd0,1
```

```
; 16 entr,es  
##16:bi0=e62,0  
; 16 sorties  
##16:o0=a62,0
```

Définition des temporisations

*Définition des E/S
(modifier suivant la configuration
de l'automate)*

Les fichiers « .SRT » et « .END »

Ces fichiers permettent d'insérer du code ABB au sein d'une application AUTOMGEN⁴.

Le code contenu dans le fichier « .SRT » est exécuté à chaque cycle, avant le code généré par la compilation de l'application AUTOMGEN.

Le code contenu dans le fichier « .END » est exécuté à chaque cycle, après le code généré par la compilation de l'application AUTOMGEN.

Le fichier « DEFABB.SRT » est fourni en modèle dans le répertoire où est installé AUTOMGEN. Ce fichier doit au minimum réaliser l'initialisation des variables de l'application. L'état du bit 0 d'AUTOMGEN est géré dans ce fichier, il doit obligatoirement être à l'état 1 uniquement au premier cycle de scrutation de l'application.

Le fichier « DEFABB.END » est fourni en modèle dans le répertoire où est installé AUTOMGEN. Ce fichier doit au minimum contenir l'instruction « !PE » (fin de programme).

Le fichier modèle « DEFABB.SRT »

Détail du fichier « DEFABB.SRT » présent dans le répertoire où est installé AUTOMGEN.

```
; Fichier .SRT pour automate ABB
```

```
; fabrique un bit toujours vrai  
!n_true=_b0_  
!n_true=_s_true_
```

*_b0_ doit être vrai uniquement
premier cycle. _true_ doit rester
toujours vrai*

⁴il est possible d'associer du code ABB à une étape Grafcet : voir le chapitre « Techniques avancées »

CHAPITRE N°6.

Syntaxe du langage ABB intermédiaire

La syntaxe est équivalente à celle du langage ABB sous forme de mnémonique.

Les paramètres des instructions sont écrits à la suite de celles-ci et séparés par un espace.

Les compléments de syntaxe suivants sont utilisables :

- pour faire référence à une variable AUTOMGEN il faut utiliser la syntaxe suivante :

`_variable_`

Par exemple : « !_m200=_m201_ » charge le contenu du mot AUTOMGEN 200 dans le mot 201.

Attention : ne pas confondre cette syntaxe avec l'utilisation des symboles du fichier « .SYM ».

- la syntaxe suivante est utilisée pour les labels de saut :

`@nom du label`

- la syntaxe suivante permet de fixer la valeur d'une constante.

`$Kn,m=valeur` (pour une constante booléenne)

`$KMn,m=valeur` (pour une constante 16 bits)

`$KDn,m=valeur` (pour une constante 32 bits)

« n,m » est le numéro de la constante,

« valeur » est une valeur de 16 bits exprimée en décimal.

Exemple :

`$KM0,0=50`

`$KD0,1=100000`

Pour comprendre l'utilisation de ces différentes syntaxes, nous vous conseillons d'observer les fichiers portant l'extension « .ABB » générés par le post-processeur.

CHAPITRE N°7.

Paramétrage de l'automate

Pour paramétrer l'automate, il faut modifier le fichier texte « .VSY » de l'application. Le fichier « DEFABB.VSY » est utilisé s'il n'y a pas de fichier « .VSY » associé à l'application. Des directives de compilation conditionnelle permettent de sélectionner des sections de programme qui seront compilées en fonction du type d'automate. La syntaxe du début de condition est : « \$IF CPU=type d'automate » et se termine par « \$ENDIF ».

Directive de sélection du type de variable utilisé par l'automate

\$CPU=

Valeurs possibles : AC ou CS.

Directive de sélection du type d'automate

;£AC31

Valeurs possibles : la présence de « £ » indique un automate AC31 ; son absence indique un automate CS31.

CHAPITRE N°8.

Compatibilité des applications avec le logiciel de programmation ABB

Les applications générées par AUTOMGEN et le post-processeur ABB peuvent être relues sous forme de liste d'instructions par le logiciel de programmation d'ABB.

CHAPITRE N°9.

Les directives de génération de code

Le générateur de code peut recevoir des directives permettant d'influer sur la forme du code généré. Ces directives sont à écrire dans le fichier « .VSX » de l'application.

Une seule directive peut être écrite par ligne, la forme est la suivante : « ;£x » où x est la directive de compilation à écrire.

;£E : gestion de l'état des étapes dans le temps à la charge du programmeur

Cette directive demande au générateur de code de ne pas générer le code d'évolution des étapes dans le temps. Ce code consiste à recopier les bits bx dans les bits x. Si cette directive est présente, alors il faut écrire le code d'évolution des étapes dans le fichier « .END ».

Cette directive peut être utile pour optimiser le programme généré en réécrivant un code de copie des bits bx vers x plus efficace (copie de table de bits).

CHAPITRE N°10.

Le module de dialogue

Le module de dialogue assure le téléchargement des programmes, la visualisation dynamique, le forçage des variables et la modification du mode de marche de l'automate.

Les types de variables suivants sont visualisables et modifiables (entrées, sorties, bits, mots, longs, consignes de temporisations pour les automates CS31 ou la valeur courante pour les automates AC31).

Le module de dialogue émule l'accès à l'état de certaines variables Système d'AUTOMGEN. L'accès en lecture est le seul autorisé pour ces variables. L'accès par programme est interdit.

- b11 : état RUN/STOP,
- b9 : contient le résultat de l'équation $M255,11 + M255,12 + M255,13$ (erreur)
- b10 : contient l'état du bit M255,14 (warning),
- b8 : chien de garde (M255,13 à 1 et MW255,0 égal à 200).

Réglage du langage utilisé pour dialoguer avec l'automate ABB

Par défaut le langage utilisés sur la prise console de l'automate ABB est l'allemand. Le module de dialogue ne fonctionne qu'avec ce langage. Si la langue anglaise a été sélectionnée, le module de communication ne pourra se connecter à l'automate. Pour sélectionner la langue allemande, procéder comme suit :

- lancer l'émulateur de terminal « ABBTERM.EXE »,
- dans le menu « PORT », choisir COM1 ou COM2,
- taper la commande « KONFS » suivi de la touche « Entrée »,
- la langue courante s'affiche, si c'est « DEUTCH » tout est correct, quitter ABBTERM, sinon presser la touche « ALT » et la laisser enfoncée, taper sur les touches « 0 », puis « 1 », puis « 2 », puis « 7 » et relâcher la touche « ALT », la langue « DEUTCH » doit maintenant s'afficher.
- appuyer sur la touche « Entrée » pour valider la nouvelle langue.

Diagnostiquer les erreurs d'exécution

Si l'automate est en erreur la consultation des mots et des bits de statut permet de connaître les causes exactes.

La table ci-dessous donne la correspondance entre les variables AUTOMGEN et les bits et les mots de statut de l'automate.

Mots AUTOMGEN	Mots de statut de l'automate
M10 à M25	MW254,0 à MW254,15
M42 à M49	MW 255,0 à MW255,7
M66 à M73	MW255,8 à MW255,15
B1 à B5	M255,10 à M255,14

CHAPITRE N°11.

Techniques avancées

Ce chapitre détaille l'utilisation de techniques sophistiquées permettant de générer des applications plus performantes.

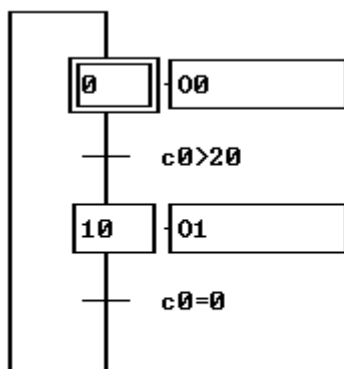
Insertion de morceaux de programmes écrits en langage ABB dans une application AUTOMGEN

Comme nous l'avons vu dans un chapitre précédent, les fichiers « .SRT » et « .END » associés à une application, peuvent recevoir des lignes de code en langage ABB.

Une autre technique peut être utilisée pour insérer du code ABB dans une boîte de code AUTOMGEN.

Les directives « #BEGIN_MACHINE_CODE » et « #END_MACHINE_CODE » permettent de définir une section de code constructeur.

Exemple :



1000	<pre>#BEGIN_MACHINE_CODE !BA0 ZUDKW #1 _m200_ ; incrément !BA0 ZUDKW #30 _m201_ ; valeur sélectionnée par i2 !BA0 URZ _i3_ ; raz du compteur _m200_ ; incrément _i2_ ; place 30 dans le compteur _m201_ ; contient 30 _true_ ; toujours validé _i0_ ; entrée de comptage _i1_ ; entrée de décomptage _c0_ ; le compteur #END_MACHINE_CODE</pre>
------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Cette technique est très intéressante pour utiliser les modules d'extension spécifiques (E/S analogiques et communication).

Références aux symboles du fichier .SYM dans les sections écrites en langage constructeur

Syntaxe :

|nom du symbole|

Le caractère « | » est généralement associé à la touche 6 du clavier.

Exemple :

Déclaration dans le fichier .SYM

niveau haut:i0

ouvrir vanne:O0

Utilisation dans les fichiers .SRT, .END ou les sections de code constructeur :

!N_|niveau haut|_=_|ouvrir vanne|_

Utilisation des tâches rapides sur les automates AC31

La directive « #I1 », « #I2 » ou « #I10 » placée en commentaire sur un folio permet d'associer le code littéral ou le code ABB écrit sur le folio à la tâche rapide 1, 2 ou cyclique. Le folio ne peut contenir qu'une seule directive « #I XX » et uniquement du code littéral bas niveau ou du code ABB écrit dans un rectangle d'organigramme.

Exemple : le but est d'inverser une sortie en fonction de la tâche rapide validée. La sortie O0 est activé au démarrage. La sortie O1 est inversée par un niveau haut sur l'entrée I7 et par une demande d'interruption de la tâche rapide 1 (front montant sur l'entrée I2). La sortie O2 est inversée par une demande d'interruption de la tâche rapide 2 (front montant sur l'entrée I3). La sortie O3 est inversée et le mot _M200_ est incrémenté toute les 2 secondes (car la condition de validation est toujours réalisée).

1- Paramétrage des conditions de validation des tâches rapides dans le fichier « .SRT » de l'application (le fichier DEFABB.SRT si l'application ne possède pas son propre fichier). Le paramétrage suivant est à écrire :

Condition de validation de la tâche rapide 1

\$IRQ1=_i7_

Condition de validation de la tâche rapide 2

\$IRQ2=_true_

Condition de validation de la tâche rapide cyclique

\$IRQ10=_true_

Période de l'interruption cyclique

\$IRQCYCLVAL=2000

2- Programmation

Le programme est composé de quatre folios : « Interruptions.GR7 » pour le programme principal, « it1.GR7 » pour la tâche rapide 1, « it2.GR7 » pour la tâche rapide 2, « it3.GR7 » pour la tâche rapide cyclique.

Programme principal : **#L"it1"**



Tâche rapide 1 :

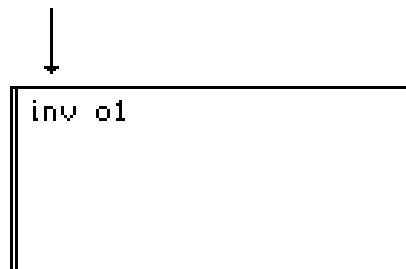
Interruption 1: front montant sur entrée I2

#L"it2"

#X"interruptions"

#I1

#P"interruptions"



Tâche rapide 2 :

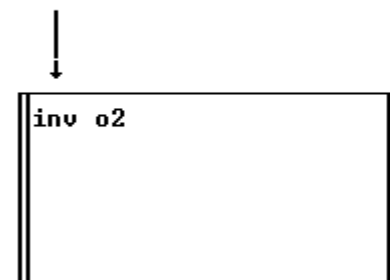
Interruption 2: front montant sur entrée I3

#X"it1"

#I2

#P"interruptions"

#L"it3"



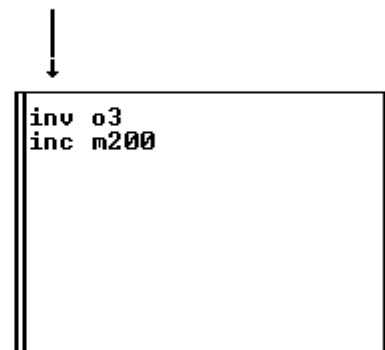
Tâche rapide cyclique :

Interruption cyclique:

#X"it2"

#P"interruptions"

#I10



Cet exemple se trouve dans le sous-répertoire « EXABB » du répertoire où est installé AUTOMGEN. Il porte le nom « Interruptions ».

CHAPITRE N°12.

Limite de compatibilité

Ce chapitre détaille les limites d'utilisation du post-processeur ABB par rapport aux possibilités d'AUTOMGEN et des automates ABB.

- les sous-programmes et les tâches rapides ne sont pas utilisables par les automates CS31,
- les doubles mots ne sont pas utilisables,
- les calculs en virgule flottante ne sont pas utilisables,
- certaines instructions du langage littéral bas niveau ne sont pas utilisables : RFZ, RFO, RFS, RFZ ainsi que toutes les instructions relatives aux calculs en virgule flottante.

Si le générateur de code ABB rencontre une forme de programme intraduisible, alors un message d'erreur de type « combinaison d'instruction et/ou de type d'adressage non supporté à l'adresse XXXX » est généré. XXXX représente une adresse dans le fichier « .EQU » Si vous souhaitez connaître avec précision l'instruction qui a généré cette erreur, utilisez l'utilitaire « CODELIST.EXE⁵ » pour obtenir un listing du fichier « .EQU ».

⁵le chapitre « Techniques avancées » du manuel de l'utilisateur associé au module principal d'AUTOMGEN décrit l'utilisation de l'utilitaire « CODELIST.EXE »

CHAPITRE N°13.

Utilisation des E/S analogiques sur une extention de l'automate AC31

Pour pouvoir utiliser les sorties analogiques sur un automate AC31 il faut :

- associer une ou plusieurs variables AUTOMGEN aux mots d'entrées sorties ABB (EW et AW) dans le fichier « .VUS » de l'application (le fichier DEFABB.VUS si l'application ne possède pas son propre fichier).

Exemple :

- automate AC31 avec une extention analogique XM06B5 utilisant 4 entrées analogiques configurables et 2 sorties analogiques configurables.
- le programme recopiera simplement l'état de la première entrée analogique (configurée en tension) sur la première sortie analogique (configurée en tension). Il permettra également un affichage spécifique de la valeur analogique de sortie. L'affichage sera initialisé ainsi : pas de décalage (OFFSET), la valeur affichée est exprimée en volt avec 2 chiffres après la virgule (coefficient multiplicateur de 10 000 et DOT=2).
- Association de variables AUTOMGEN aux mots d'entrées / sorties ABB : dans le fichier « .VUS » de l'application : à la fin du fichier ajoutez les lignes suivantes :

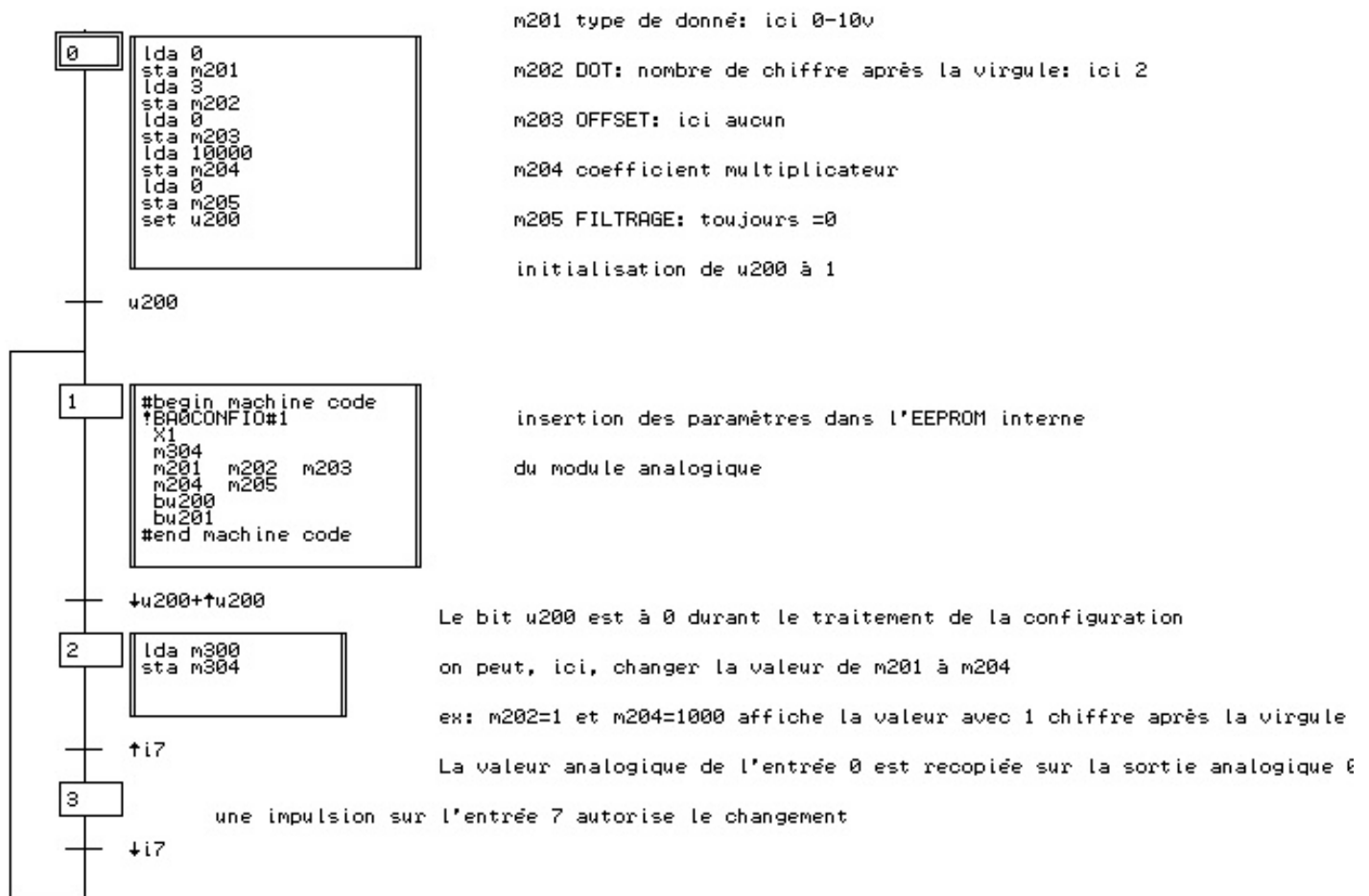
```
##4:M300=EW63,0
```

```
##2:M304=AW63,0
```

Ces deux lignes associent les mots M300 à M303 d'AUTOMGEN aux variables EW63,0 à EW63,3 de l'automate ainsi que les variables M304 et M305 d'AUTOMGEN aux variables AW63,0 et AW63,1 de l'automate.

1- programmation :

CONFIGURATION DE L' AFFICHAGE DES ENTREES SORTIES ANALOGIQUES POUR L'EXTENSION XM06B5 DE L'AUTOMATE AC31



Cet exemple se trouve dans le sous-répertoire « EXABB » du répertoire où est installé AUTOMGEN. Il porte le nom « Analogique ».

CHAPITRE N°14.

Exemple complet

Ce chapitre contient le dossier de documentation de l'exemple « ABB ».

Cet exemple est présent dans le sous-répertoire « EXABB » du répertoire où est installé AUTOMGEN.

ABB

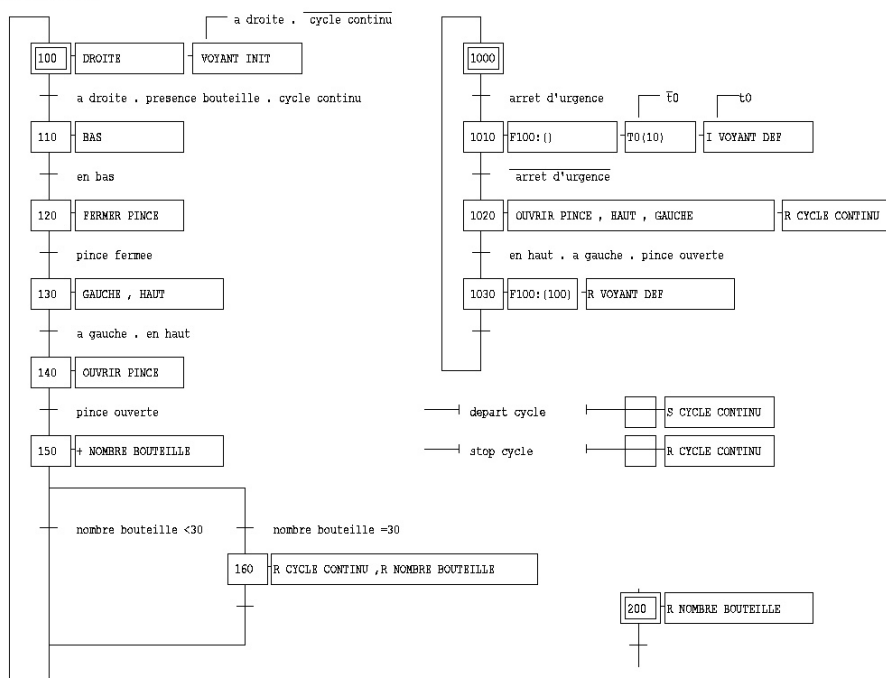


IRAI

ABB

Date de création : 09/12/1999 15:49 00

* Programme d'exemple *



AUTOMGEN	APT N°1	APT N°2	SYMBOLS	COMMENTAIRE OU NOM DES FOLIOS OU LA VARIABLE EST UTILISEE
I0000		E62,0	a droite	capteur verin 1 sorti
I0001		E62,1	a gauche	capteur verin 1 rentre
I0002		E62,2	en bas	capteur verin 2 sorti
I0003		E62,3	en haut	capteur verin 2 rentre
I0004		E62,4	pince fermee	capteur pince fermee
I0005		E62,5	pince ouverte	capteur pince ouverte
I0006		E62,6	presence bouteille	capteur bouteille en attente
I0007		E62,7	depart cycle	bp depart de cycle
I0008		E62,8	stop cycle	bp stop
I0009		E62,9	arret d'urgence	bp arret d'urgence
M0200	MW0,2		nombre bouteille	nombre de bouteilles passees
O0000	A62,0		DROITE	sortir le verin 1
O0001	A62,1		GAUCHE	rentrer le verin 1
O0002	A62,2		BAS	sortir le verin 2
O0003	A62,3		HAUT	rentrer le verin2
O0004	A62,4		FERMER PINCE	fermer la pince
O0005	A62,5		OUVRIR PINCE	ouvrir la pince
O0006	A62,6		VOYANT INIT	voyant vert
O0007	A62,7		VOYANT DEF	voyant rouge
T0000	M2,5	M2,13		
U0101	M1,14	M1,15		
U0103	M1,12	M1,13		
U1000	M1,14	M1,15	cycle continu	etat cycle continu
X0100	M0,4	M0,5		
X0110	M0,6	M0,7		
X0120	M0,8	M0,9		
X0130	M0,10	M0,11		
X0140	M0,12	M0,13		
X0150	M0,14	M0,15		
X0160	M1,0	M1,1		
X0200	M1,2	M1,3		
X1000	M1,4	M1,5		
X1010	M1,6	M1,7		
X1020	M1,8	M1,9		
X1030	M1,10	M1,11		
	M1,10	M1,11		
<p>ABB - Page 1 - La liste des références croisées</p>				

Fichier « ABB.ABB »

```
; ***** Variables systèmes ***** VSY
; pas de fichier C:\AUTOMV6\EXABB\Abb.VSY pour l'application, utilisation du fichier
C:\AUTOMV6\defabb.VSY
; Fichier .VSY pour automate ABB

#true=m0,0      ; un bit toujours vrai (fabriqué dans le fichier SRT)
#iv=m0,1        ; un bit pour évaluation intermédiaire
#it=m0,2        ; un bit pour évaluation intermédiaire
#b0=m0,3        ; un bit vrai au premier cycle
#iw=mw0,0       ; un mot pour évaluation intermédiaire
#ac=mw0,1       ; accumulateur 16 bits
#dw=md0,0       ; un mot double pour évaluation intermédiaire
#la=md0,1       ; accumulateur 32 bits

; pour le traitement des erreurs
##5:b1=m255,10
##5:bb1=m255,10

##16:m10=mw254,0
##8:m42=mw255,0
##8:m66=mw255,8

; configuration de l'automate

$KW0,0=-2      ; stand alone central unit
; ***** Variables utilisateur ***** VUS
; pas de fichier C:\AUTOMV6\EXABB\Abb.VUS pour l'application, utilisation du fichier
C:\AUTOMV6\defabb.VUS
; Fichier .VUS pour automate ABB

; bits (m0,0 à m0,3 sont réservés dans le .VSY)
###i&bo&x&bx&b&bb&u&bu&t&bt&ui&di&uo&do&ux&dx&uu&du&ub&db&tt&ut&dt=m0,4:21,15
###i&bo&x&bx&b&bb&u&bu&t&bt&ui&di&uo&do&ux&dx&uu&du&ub&db&tt&ut&dt=m230,0:239,15

; mots et compteurs (mw0,0 à mw0,1 sont réservés dans le .VSY)
###c&m=mw0,2:5,15
###c&m=mw230,0:239,15

; doubles mots (dw0,0 à dw0,1 sont réservés dans le .VSY)
###l=md0,2:1,15

; 16 temporisations
##16:tconsi0=kd0,1

; 16 entrées
##16:bi0=e62,0

; 16 sorties
##16:o0=a62,0
; ***** Correspondance bits/étapes
#b100=_x100_
#bb100=_bx100_
#b104=_x110_
#bb104=_bx110_
#b106=_x120_
#bb106=_bx120_
#b108=_x130_
#bb108=_bx130_
#b110=_x140_
#bb110=_bx140_
#b113=_x150_
#bb113=_bx150_
#b116=_x160_
#bb116=_bx160_
#b117=_x200_
#bb117=_bx200_
#b102=_x1000_
#bb102=_bx1000_
#b105=_x1010_
#bb105=_bx1010_
#b107=_x1020_
#bb107=_bx1020_
#b109=_x1030_
#bb109=_bx1030_
; ***** Correspondance bits/bits utilisateur
#b103=_u100_
#bb103=_bu100_
#b101=_u1000_
#bb101=_bu1000_
```



```

; ***** Programme de démarrage ***** SRT
; pas de fichier C:\AUTOMV6\EXABB\Abb.SRT pour l'application, utilisation du fichier
C:\AUTOMV6\defabb.SRT
; Fichier .SRT pour automate ABB

; fabrique un bit toujours vrai
!n_true=_b0_
!n_true=_s_true_

; ***** Prédispositions
!N_b0=_iv_
!BA0SPRUNG_iv_label10_#0#0
$_tconsi0_=1000
@label10
; ***** Code généré par le post-processeur *****

; ***** Module : C:\AUTOMV6\EXABB\ABB.GR7
!_bi0_&_bi6_&_u1000_=R_bx100_
!_b0_=S_bx100_
!_xl60_=_bu99_
!_xl50_=_iv_
!BA0ZUDKW#30_iw_
!_m200_<_iw=_it_
!_iv_&_it=_bu98_
!_bu99_/_bu98_=S_bx100_
!_u100_=R_bx1000_
!_b0_=S_bx1000_
!_xl030_=S_bx1000_
!_bi2_=R_bx110_
!_xl00_&_bi0_&_bi6_&_u1000_=S_bx110_
!N_u100_=R_bx1010_
!_xl000_&_u100_=S_bx1010_
!_bi4_=R_bx120_
!_xl10_&_bi2_=S_bx120_
!_bi3_&_bi1_&_bi5_=R_bx1020_
!_xl010_&N_u100_=S_bx1020_
!_bi1_&_bi3_=R_bx130_
!_xl20_&_bi4_=S_bx130_
!_true_=R_bx1030_
!_xl020_&_bi3_&_bi1_&_bi5_=S_bx1030_
!_bi5_=R_bx140_
!_xl30_&_bi1_&_bi3_=S_bx140_
!_bi7_=_b111_
!BA0ZUDKW#30_iw_
!_m200_=?_iw=_it_
!_it=_iv_
!BA0ZUDKW#30_iw_
!_m200_<_iw=_it_
!_iv_/_it_=R_bx150_
!_xl40_&_bi5_=S_bx150_
!_bi8_=_b114_
!_true_=R_bx160_
!_xl50_=_iv_
!BA0ZUDKW#30_iw_
!_m200_=?_iw=_it_
!_iv_&_it_=S_bx160_
!_true_=R_bx200_
!_b0_=S_bx200_
!_bi0_&N_u1000_=_b119_
!N_t0=_b122_
!_t0=_b124_
!N_b120=_iv_
!BA0SPRUNG_iv_line99_#0#0
!_true_=R_bx140_
!_true_=R_bx130_
!_true_=R_bx120_
!_true_=R_bx110_
!_true_=R_bx100_
!_true_=R_bx160_
!_true_=R_bx150_
@line99
!N_b125=_iv_
!BA0SPRUNG_iv_line108_#0#0
!_true_=R_bx140_
!_true_=R_bx130_
!_true_=R_bx120_
!_true_=R_bx110_
!_true_=S_bx100_
!_true_=R_bx160_
!_true_=R_bx150_
@line108
; ***** Module : (ACTION)
!_b119_&_bx100_=_b118_
!_b122_&_bx1010=_b121_
!_b124_&_bx1010=_b123_

```

```

!_bx100=_o0_
!_b118=_o6_
!_bx110=_o2_
!_bx1010=_b120_
!_b121=_bt0_
!_b123=_N_iv_
!BA0SPRUNG_iv__label1_#0#0
!N_o7=_o7_
@label1
!_bx120=_o4_
!_bx1020/_bx140=_o5_
!_bx1020/_bx130=_o3_
!_bx1020/_bx130=_o1_
!_bx1020/_b114/_bx160=R_bu1000_
!_bx1030=_b125_
!_bx1030=R_o7_
!_b111=S_bu1000_
!_bx150=_N_iv_
!BA0SPRUNG_iv__label2_#0#0
!BA0ZUDKW#1_iw_
!_m200+_iw=_m200_
@label2
!_bx160/_bx200=_N_iv_
!BA0SPRUNG_iv__label3_#0#0
!BA0ZUDKW#0_m200_
@label3
; ***** Evolution des bits d'étapes
!_bx100=_x100_
!_bx110=_x110_
!_bx120=_x120_
!_bx130=_x130_
!_bx140=_x140_
!_bx150=_x150_
!_bx160=_x160_
!_bx200=_x200_
!_bx1000=_x1000_
!_bx1010=_x1010_
!_bx1020=_x1020_
!_bx1030=_x1030_
; ***** Evolution des bits utilisateurs
!_bu100=_u100_
!_bu1000=_u1000_
; ***** Evolution des temporisations
!BA0ESV_bt0__tconsi0__t0_
!N_bt0=R_t0_
; ***** Programme de fin
; ***** END
; pas de fichier C:\AUTOMV6\EXABB\Abb.END pour l'application, utilisation du fichier
C:\AUTOMV6\defabb.END
; Fichier .END pour automate ABB

!pe

```

Fichier « ABB.LAP »

Code ABB généré par AUTOMGEN

```
!NM0,0=M0,3
!NM0,0=SM0,0
!NM0,3=M0,1
!BA0SPRUNGM0,1MA0#0#0
MA0

; MODULE : C:\AUTOMV6\EXABB\ABB.GR7

!E62,0&E62,6&M1,14=RM0,5
!M0,3=SM0,5
!M1,0=M2,0
!M0,14=M0,1
!BA0ZUDKW#30MW0,0
!MW0,2<MW0,0=M0,2
!M0,1&M0,2=M2,1
!M2,0/M2,1=SM0,5
!M1,12=RM1,5
!M0,3=SM1,5
!M1,10=SM1,5
!E62,2=RM0,7
!M0,4&E62,0&E62,6&M1,14=SM0,7
!NM1,12=RM1,7
!M1,4&M1,12=SM1,7
!E62,4=RM0,9
!M0,6&E62,2=SM0,9
!E62,3&E62,1&E62,5=RM1,9
!M1,6&NM1,12=SM1,9
!E62,1&E62,3=RM0,11
!M0,8&E62,4=SM0,11
!M0,0=RM1,11
!M1,8&E62,3&E62,1&E62,5=SM1,11
!E62,5=RM0,13
!M0,10&E62,1&E62,3=SM0,13
!E62,7=M2,2
!BA0ZUDKW#30MW0,0
!MW0,2=?MW0,0=M0,2
!M0,2=M0,1
!BA0ZUDKW#30MW0,0
!MW0,2<MW0,0=M0,2
!M0,1/M0,2=RM0,15
!M0,12&E62,5=SM0,15
!E62,8=M2,3
!M0,0=RM1,1
!M0,14=M0,1
!BA0ZUDKW#30MW0,0
!MW0,2=?MW0,0=M0,2
!M0,1&M0,2=SM1,1
!M0,0=RM1,3
!M0,3=SM1,3
!E62,0&NM1,14=M2,4
!NM2,5=M2,6
!M2,5=M2,7
!NM2,8=M0,1
!BA0SPRUNGM0,1MA1#0#0
!M0,0=RM0,13
!M0,0=RM0,11
!M0,0=RM0,9
!M0,0=RM0,7
!M0,0=RM0,5
!M0,0=RM1,1
!M0,0=RM0,15
MA1
!NM2,9=M0,1
!BA0SPRUNGM0,1MA2#0#0
!M0,0=RM0,13
!M0,0=RM0,11
!M0,0=RM0,9
!M0,0=RM0,7
!M0,0=SM0,5
!M0,0=RM1,1
!M0,0=RM0,15
MA2

; MODULE : (ACTION)

!M2,4&M0,5=M2,10
!M2,6&M1,7=M2,11
!M2,7&M1,7=M2,12
!M0,5=A62,0
!M2,10=A62,6
```

```

!M0,7=A62,2
!M1,7=M2,8
!M2,11=M2,13
!M2,12=NM0,1
!BA0SPRUNGM0,1MA3#0#0
!NA62,7=A62,7
MA3
!M0,9=A62,4
!M1,9/M0,13=A62,5
!M1,9/M0,11=A62,3
!M1,9/M0,11=A62,1
!M1,9/M2,3/M1,1=RM1,15
!M1,11=M2,9
!M1,11=RA62,7
!M2,2=SM1,15
!M0,15=NM0,1
!BA0SPRUNGM0,1MA4#0#0
!BA0ZUDKW#1MW0,0
!MW0,2+MW0,0=MW0,2
MA4
!M1,1/M1,3=NM0,1
!BA0SPRUNGM0,1MA5#0#0
!BA0ZUDKW#0MW0,2
MA5
!M0,5=M0,4
!M0,7=M0,6
!M0,9=M0,8
!M0,11=M0,10
!M0,13=M0,12
!M0,15=M0,14
!M1,1=M1,0
!M1,3=M1,2
!M1,5=M1,4
!M1,7=M1,6
!M1,9=M1,8
!M1,11=M1,10
!M1,13=M1,12
!M1,15=M1,14
!BA0ESVM2,13KD0,1M2,5
!NM2,13=RM2,5
!PE

```