HUITIEME COLLOQUE SUR LE TRAITEMENT DU
SIGNAL ET SES APPLICATIONS

NICE du 1er au 5 JUIN 1981

AUTOMATE GRAFCET

R. BORRIN, N. HARTENSTEIN, J. RANCHIN

CNAM 2 rue Conté - 75003 PARIS

RESUME

Un automate programmable de haut de gamme a été réalisé autour d'un microprocesseur 8085, il est muni de 40 K de mémoire, de 2 disquettes et d'entrées/sorties industrielles. Un langage d'expression du GRAFCET a été défini qui permet l'expression en format libre et qui fournit des sorties étiquettes. Le compilateur a été rédigé en Assembleur 8085. Cet automate s'utilise via 2 programmes :

- un programme maître qui gère l'automatisme, fait l'acquisition des informations d'entrée, appelle le programme utilisateur (compilé), envoie en retour les ordres d'action,
- le programme utilisateur qui exécute l'automate.

Le compilateur a été réalisé à l'aide de matrices de transition pour obtenir une plus grande rapidité d'exécution. De par sa conception cet automate est un ordinateur complet, le système d'exploitation de disquettes permet des changements de programme (changement d'automatisme), l'archivage des informations d'entrée et de sortie, le fonctionnement en simulation.

SUMMARY

A high level programmable automaton has been built with 8085 CPU, 40 K byte memory, 2 floppy disks and industrial I/O.
A Grafect language has been designed. It permits free expression and gives in return labelled outputs.
Compilator is written in assembly language of 8085 CPU.

One uses the automaton by two programs :

- master program controls the external process, industrial input and output and call the user program,
- user program drives the automaton.

The compiler uses transition matrices for high speed execution.
The automaton is a complete computer, DOS gives ability of changing programs, storage of I/O data and simulations.

AUTOMATE GRAFCET

INTRODUCTION

L'apparition des microprocesseurs, les prix de revient en constante diminution et la taille réduite des appareils favorisent l'exploitation des mini et des micro-ordinateurs. Nous pouvons espérer que cet avènement ouvrira de nouvelles perspectives dans la conduite de processus et que les automates programmables vont se substituer aux automates cablés. Les inconvénients de ces derniers se situent à deux niveaux principaux :

- encombrement du matériel,
- caractère figé des réalisations, ce qui a pour conséquence d'accroître les difficultés de modification et de réalisation en raison du coût élevé du cablage et des études.

Les automates programmés pallient ces inconvénients. Le laboratoire de micro-informatique du C.N.A.M. s'est donc intéressé dans un premier temps à l'implantation d'un automate programmable - baptisé AP 85 - autour d'un microprocesseur 80 85 de chez INTEL.

Simultanément, et consécutivement à cette apparition des microprocesseurs, le coût de la conception va en s'accroissant. Aussi l'objectif qui se dégage actuellement est-il d'optimiser les temps d'étude et d'analyse. Sensible à cette évolution, le laboratoire s'est donc préoccupé des méthodes nouvelles de représentation du cahier des charges - Réseaux de Pétri et Grafcet - qui présentent l'avantage :

- d'être graphiques,
- de prendre en compte les évolutions parallèles,
- de ne considérer à un instant donné que les variables significatives,
- d'être indépendantes de toute technologie.

L'automate programmable AP 85 et son logiciel permettant :

- des traitements complexes pouvant porter sur des valeurs de compteurs,
- une gestion de fichiers facilitant le stockage de programmes,

nous pouvons classer ce matériel parmi le "haut de gamme". Il était donc intéressant d'adjoindre à ce micro-ordinateur industriel réalisé au laboratoire, un compilateur d'un langage spécialisé dans la conception d'automates. L'étude de faisabilité et la réalisation de ce compilateur devait tenir compte de la vocation du laboratoire pour l'enseignement en sus de sa fonction de recherche. Nos objectifs étaient donc de développer un langage permettant :

- la réalisation de petits automatismes,
- une conception assistée par ordinateur.

L'utilisateur, même non informaticien, peut ainsi espérer établir un programme aboutissant à une réalisation optimale, c'est-à-dire présentant l'enchaînement le plus avantageux des tâches. Ceci nous a conduit à réaliser un compilateur conçu pour un langage simple, en français, et directement applicable à partir du mode de représentation choisi : le Grafcet de niveau deux c'est-à-dire présentant les spécifications technologiques et opérationnelles de l'automatisme à réaliser.

I - L'AUTOMATE PROGRAMMABLE AP 85

1 - ARCHITECTURE DE L'AUTOMATE :

L'automate programmable a été conçu autour du microprocesseur huit bits (8085) auquel sont associés

- 4 K octets de mémoire morte,
- 40 K octets de mémoire vive statique,
- un contrôleur de disques souples,
- des interfaces pour :
  - le contrôle des entrées - sorties sur le télétype ou l'écran,
  - la transmission sur imprimante,
  - le contrôle des entrées - sorties avec le processus par l'intermédiaire des capteurs et des actionneurs ,
- un contrôleur d'interruption pour la gestion des interruptions extérieures au système,
- un circuit de temporisation gérant trois compteurs programmables de seize bits chacun,
- deux convertisseurs analogiques - numériques de huit bits.

Schéma d'Architecture ci-après.

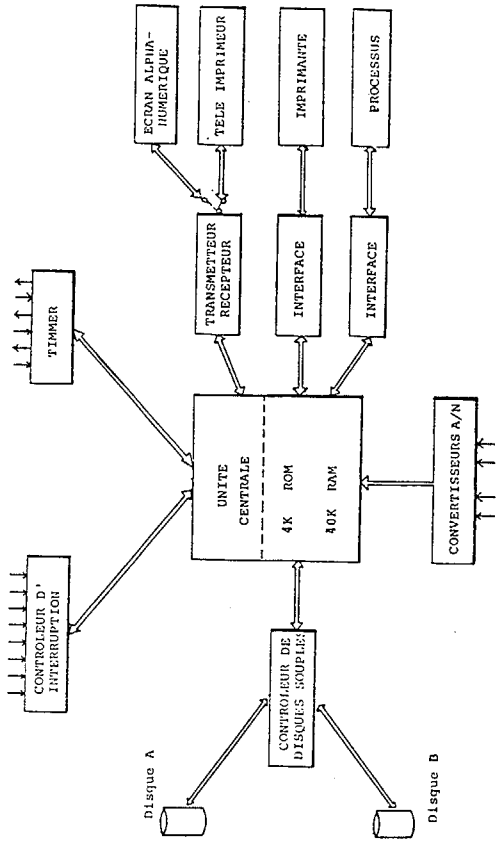
2 - LE MONITEUR :

Le moniteur d'exploitation a pour objet :

- de vérifier la frappe d'une commande
- et de faciliter l'exécution des tâches demandées.

Diverses fonctions sont exécutables à partir du moniteur d'exploitation sur disques :

- l'édition d'un ou plusieurs secteurs (fonction "DUMP")
- chargement et exécution d'un programme avec des points d'arrêt ("DEBUG")
- édition du répertoire ("DIR")
- dédoublage d'un fichier ("RECOP")
- sauvegarde d'un fichier sur ruban perforé ("PUNCH")
- édition d'un fichier ("LIST" ou "PRINT")
- création d'un fichier à partir du ruban perforé ("CREAT")



ARCHITECTURE DE L'AUTOMATE PROGRAMMABLE

- recopie d'un ou de tous les fichiers sur d'autres disquettes ("COPY")
- test d'une disquette ("TEST")
- accès à un secteur ("DISK")
- éditeur de texte source ("EDIT")

nous allons plus particulièrement détailler les fonctions :

- d'assemblage d'un programme ("ASM")
- de chargement d'un programme ("LOAD")
- d'effacement d'un fichier ("DELET")

qui vont être indispensables à l'exploitation du compilateur que nous avons créé ; mais d'ores et déjà, nous pouvons constater que toutes les possibilités matérielles et logicielles de l'AP 85 le situe parmi les automates de haut de gamme.

effacement d'un fichier :

DELET <nom de fichier> [#<type de fichier>] (RC)

- en l'absence de précision sur le type du fichier, le système considère qu'il s'agit du fichier source.
- en présence du signe * tous les fichiers quel qu'en soit le type sont effacés.

chargement et exécution d'un ou plusieurs programmes :

LOAD <nom du fichier 1>,.....<nom fichier i> (RC)

avec 1 ≤ i ≤ 8

le contrôle est donné au premier fichier.

assemblage d'un programme :

L'assembleur permet, à partir d'un texte source écrit en langage symbolique, d'obtenir un programme exécutable

ASM <nom de fichier> <option éventuelle> (RC).

II - PRESENTATION DU LANGAGE :

A - LES INSTRUCTIONS :

1 - LES DECLARATIONS :

- ENTREES <nom d'entrée> == (Constante), [<nom d'entrée> == (Constante),.....] ;
- SORTIES <nom de sortie> == (Constante), [<nom de sortie> == (Constante),.....] ;

Le nom d'entrée et de sortie doit obligatoirement débuter par A,B ou C afin d'indiquer le port d'entrée ou de sortie. La constante peut prendre les valeurs 01-02-04-08-10-20-40 ou 80.

- ETAPES <nom d'étape> == (Constante), [<nom d'étape> == (Constante),.....] ;

La constante est égale à 1 ou 0. La valeur 1 signale la présence d'une étape initiale active.

-COMPTEUR <nom compteur> [== (Constante)] , [<nom compteur> [== (Constante)],.....] ;

La constante si elle est présente doit être inférieure à la valeur 256.

-TEMPORISATION <non temporisation> == (Constante/Constante), [<nom temporisation> == (Constante/Constante),..];

La 1ère constante indique les secondes, la 2ème constante indique les dixièmes de secondes.

-TRANSITION <non transition> == (<Etape précédente,> ...../ <étape suivante,>...), [<nom transition> == (<Etape précédente,> ...../ <étape suivante,>....) ...] ;

-ZONE Constante OCTETS <identifieur> [=("Caractères")];

2 - LES INSTRUCTIONS :

- PROGRAMME <non programme>
- PROCEDURE <identifieur> [(<identifieur>, < >,... < >)] ;
- FIN; pour signaler la fin d'une procédure ou celle d'une instruction conditionnelle.
- FIN. pour indiquer la fin du programme.
- FAIRE <nom de sortie> [DURANT <nom temporisation>];

la temporisation doit être associée à une valeur de sortie du port A.

- ANNULER <nom de sortie>;
- ALLER A <identifieur>;

l'identifieur peut être un nom d'étape, de transition ou une étiquette.

- CONTINUER ; pour signaler une transition validée.
- ARRETER ; pour retourner au moniteur d'exploitation sur disque.
- LIRE <nom de zone> ; arrêt sur RC
- ECRIRE <nom de zone> ; arrêt sur compte d'octets.
- INCR <nom de compteur> ;
- DECR <nom de compteur> ;
- APPELER <identifieur> [(identifieur , ... , ...)];
- MOUVEMENTER Constante OCTETS DE Identifieur
  Identifieur (Constante)
  (Identifieur)
  sur identifieur (Constante) ;
  (Identifieur)

-<identifieur> : =
<identifieur> [opérateur <constante>] ;
<constante> [opérateur <identifieur>]
opérateur : +,-,*,
et,ou,non

-SI (<identifieur> opérateur 1
<identifieur>) opérateur 2()
<constante>
ALORS instruction ;
SINON instruction ;
opérateur 1: <,<=,>, >=, /=,=
opérateur 2: ET, OU (le ET est prioritaire).

Toutes les instructions ainsi que les déclarations doivent être précédées d'un caractère blanc ou ";" en colonne 1.

Toutes les instructions à l'exception des pseudo-instructions peuvent être précédées d'une étiquette.
L'instruction se présentera alors de la manière suivante :

<identifieur> : instruction ;

où l'identifieur possède huit caractères au maximum avec une lettre comme caractère initial de même que tous les identifieurs présents dans les déclarations ou les instructions.

La présentation de l'instruction est laissée à l'initiative de l'utilisateur :
- une instruction peut être écrite sur n lignes.
- n instructions peuvent figurer sur la même ligne.

Exemples :

```
TRANSITIONS T1 = (E1, E2),
            T2 = (E2, E3),

            T8 = (ES/E1);
SI (BV2 = 02) ALORS CONTINUER ; FIN ;
```

Le caractère ";" signale la fin d'une instruction.

3 - LES ERREURS DE SYNTAXE

Le message apparaît sous la forme :
ERG : xx [n° ligne] * Symbole *
Il signifie qu'une erreur de type xx a été détectée à la ligne précisée entre crochet. Le symbole sur lequel s'arrête l'analyse est éventuellement précisé entre deux étoiles.

B - STRUCTURE GENERALE D'UN PROGRAMME :

1 - EXEMPLE DE PROGRAMME EN LANGAGE ORIENTE GRAFCET

Le programme est celui correspondant au cahier des charges présenté dans le chapitre 1.

Le programme de type "P" est constitué de la manière suivante :

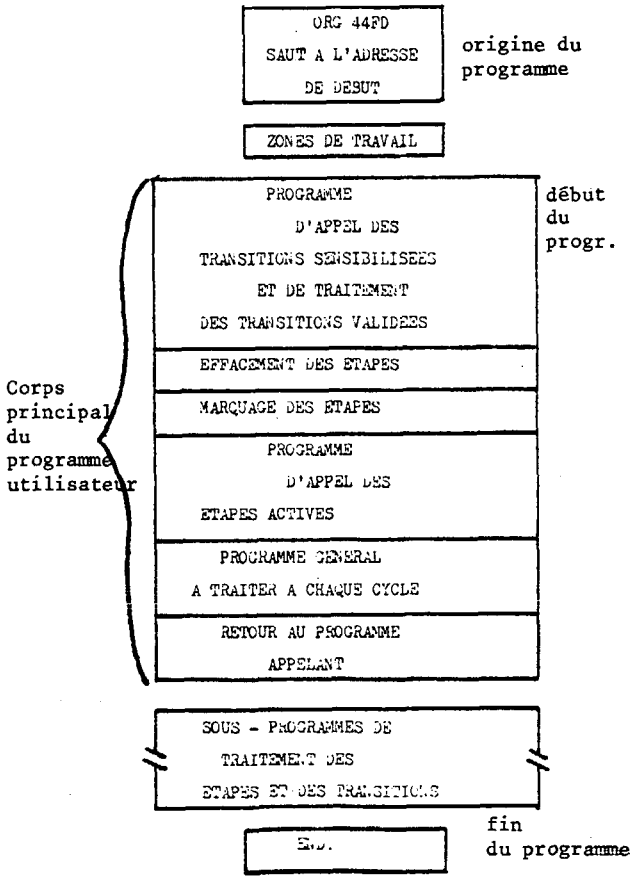
PROGRAMME <nom de programme>
DECLARATIONS
SOUS-PROGRAMMES éventuels
INSTRUCTIONS A EXECUTER A CHAQUE CYCLE
INSTRUCTIONS CORRESPONDANT AUX ETAPES ET AUX TRANSITIONS.

2 - STRUCTURE D'UN PROGRAMME SOURCE ASSEMBLEUR :

a) SCHEMA GENERAL DU PROGRAMME :

Le programme de l'utilisateur prend en compte les entrées lues par un programme interpréteur (détaillé dans la suite de ce chapitre) afin de modifier les sorties. L'ordre d'action ne sera donné qu'à l'issue

du cycle de traitement et par le programme interpréteur.



SCHEMA GENERAL DU PROGRAMME ASSEMBLEUR

b) PRINCIPE D'EXECUTION DU PROGRAMME :

Après acquisition des états des entrées, dans le programme interpréteur, seule l'évolution dans l'activation des étapes est considérée dans un premier temps.

Pour chaque transition :

- les étapes précédentes sont examinées.
- si la transition considérée n'est pas sensibilisée, nous passons au traitement de la transition suivante s'il y a lieu ; sinon nous sautons à la partie effacement des étapes actives.
- si la transition est sensibilisée, un appel est fait alors au sous-programme correspondant à cette transition. Au retour, on vérifie que la transition est validée.
- si la transition n'est pas validée, nous passons au traitement de la transition suivante s'il y a lieu ; sinon nous sautons à la partie effacement des étapes actives.
- si la transition est validée :
  * l'adresse des étapes à effacer est mémorisée,
  * l'adresse des étapes à activer l'est également dans une autre zone.

Lorsque toutes les transitions ont été traitées, on procède :

- à l'effacement réel des étapes mémorisées précédemment (étapes précédentes des transitions validées).
- à l'activation effective des étapes suivantes des transitions validées.

Si aucune transition n'avait été validée dans la phase précédente, on passe à la phase suivante.

Le mode de traitement permet la résolution des situations conflictuelles que nous avons examinées dans le chapitre 1; à savoir : le traitement des transitions qui partagent une même étape. L'effacement réel de l'étape n'ayant lieu qu'à l'issue du traitement de toutes les transitions, ces transitions peuvent être validées dans le même cycle de traitement. Ceci nécessite bien évidemment l'utilisation d'une mémoire d'étape.

Pour chaque étape :

- si l'étape est active, le sous-programme de traitement correspondant est appelé. Dans ce sous-programme sont indiqués les ordres d'activation ou d'annulation des actions. Dans le cas d'un ordre d'activation, un OU LOGIQUE est effectué entre la zone mémoire du port de sortie et la valeur de sortie considérée. L'annulation d'une action est légèrement plus complexe. On effectue un OU EXCLUSIF entre la valeur de sortie considérée et la valeur FF_H puis un ET LOGIQUE entre la zone mémoire du port de sortie et le complément à un obtenu précédemment.

A l'issue de ce traitement, un traitement obligatoire à chaque fin de cycle peut être imposé. Dans le cas d'un déroulement normal (pas d'arrêt d'urgence), le retour au programme interpréteur permet l'envoi des sorties ou ordres d'activation en direction du processeur. Le cycle recommence alors par l'acquisition des nouvelles entrées.

CONCLUSION :

Nous nous sommes efforcés de garder constamment à l'esprit la finalité de notre travail :

- recherche d'un gain dans le coût de la conception,
- allié à un but d'enseignement.

La réalisation de ce langage spécialisé dans la con-

ception d'automates pouvait être envisagée soit par la création d'un interpréteur soit par celle d'un compilateur. Mais l'exploitation d'un interpréteur en milieu industriel paraissait peu vraisemblable aussi, malgré tous ses avantages nous avons choisi la réalisation d'un compilateur. Le gain dans le coût de conception a été obtenu principalement par la programmation directe du Grafcet de niveau 2 dans le langage. L'emploi de termes du Grafcet comme mots clés du langage "étapes, transitions" en français facilite son apprentissage. L'utilisation d'un compilateur d'un langage orienté Grafcet présente tous les avantages inhérents à un compilateur mais uassi tous ses inconvénients et principalement la nécessité d'une place mémoire plus importante d'une part pour l'implantation du compilateur et d'autre part pour la création du programme source assembleur.

Ce compilateur est caractérisé par ses deux entrées possibles :

- fonction de "CREATION"
- fonction de "TRADUCTION"

La fonction de création permet à l'utilisateur d'introduire son programme sous le contrôle du compilateur. Toute instruction rejetée après analyse peut, après correction de l'erreur, être proposée à nouveau. Cette possibilité va accélérer la correction des erreurs autres que celles de logique. La fonction de traduction permet, après modification, l'analyse d'un programme sans intervention de l'opérateur.

L'utilisateur de notre système ainsi que de l'automate programmable AP 85 pourrait être facilitée par :

- la création d'un logiciel procédant à l'analyse et à la vérification du Grafcet décrit ce qui en faciliterait la conception,
- l'adjonction au compilateur de commandes permettant :
  - * la visualisation des zones,
  - * l'édition des différents labels

lors du passage dans tout ou partie du programme en cours d'exécution. La mise au point de programmes complexes en serait facilitée.

- l'utilisation d'une table traçante permettant une mise à jour automatique du cahier des charges ce qui constituerait une aide précieuse pour le concepteur.
- l'emploi d'un écran graphique visualisant le processus et mettant en relief les actions en cours d'exécution.

BIBLIOGRAPHIE

"LE GRAFCET"
DIAGRAMME FONCTIONNEL DES AUTOMATES SEQUENTIELS
Agence nationale pour le Développement de la Production Automatisée
A.D.E.P.A.
Avril 1979

POUR UNE REPRESENTATION NORMALISEE DU CAHIER DES CHARGES D'UN AUTOMATISME LOGIQUE
par le groupe de travail SYSTEMES LOGIQUES DE L'AFCET
REVUE : AUTOMATIQUE ET INFORMATIQUE
Novembre - Décembre 1977

ETUDE COMPARATIVE DE DEUX OUTILS DE REPRESENTATION : GRAFCET ET RESEAU DE PETRI
R. VALETTE
REVUE : LE NOUVEL AUTOMATISME
Décembre 1978 (pages 377 à 382)

ETUDE ET REALISATION D'UN AUTOMATE PROGRAMMABLE AUTOUR D'UN MICROPROCESSEUR : AP 85
Gérard HAETTEL
Mémoire d'ingénieur CNAM
Année 1980

TECHNIQUES DE COMPILATION
méthodes d'analyses syntaxiques
H. GALLAIRE
Collection : SUP'AERO
Ecole Nationale de l'Aéronautique et de l'espace
Cépadues - éditions
Année 1977