CALCULATEUR ELECTRONIQUE WANG

LOCI

MODE D'EMPLOI

1) CARACTERISTIQUES GENERALES

Le calculateur électronique WANG, modèle LOCI II, est une machine digitale qui présente les caractéristiques suivantes :

capacité	10 chiffres et virgule	-10^{-10} à $10^{10}-1$
addition, soustraction	précision 10 chiffres	
multiplication, division, log.	précision 8 chiffres	
registres	capacité 10 chiffres	
registre logarithmique	capacité 2 chiffres	pour la caractéristique
	capacité 10 chiffres	pour la mantisse
virgule	flottante	

Il se compose essentiellement :

- d'un clavier comportant les touches permettant l'entrée des chiffres et la commande des ordres et fonctions à exécuter
- d'un registre de travail W
- d'un accumulateur A
- d'un dispositif calculant le logarithme ou l'antilogarithme
- et d'un accumulateur L

Les appareils peuvent être complétés par 2, 4 ou 16 mémoires. Ils comportent un dispositif permettant l'exécution automatique des calculs en fonction d'un programme enregistré sur une carte perforée et lu par un lecteur statique.

Les appareils peuvent être munis de circuits complémentaires avec sortie du registre W en signaux série ou parallèle, une entrée extérieure permettant le raccordement à des équipements périphériques tels que Télétype, Presin, lecteur de bande perforée, etc...

2) MISE EN SERVICE

La machine étant branchée, appuyer sur le petit disjoncteur rouge qui se trouve à l'arrière, en bas à droite, attendre 4 à 5 minutes et l'ensemble est alors prêt à fonctionner (pour déclencher, il suffit d'appuyer sur la partie inférieure de ce même disjoncteur afin de la déverrouiller). Le ventilateur se met en marche. Mettre l'inverseur à droite du clavier en position "STEP" ou "MANUEL", appuyer sur la touche "PRIME" et éteindre éventuellement le voyant "ERREUR" à l'aide de la touche "CL ER".

3) SIGNIFICATION DES TOUCHES, VOYANTS ET COMMUTATEURS

Remarque : Les claviers des LOCI peuvent différer selon les modèles et les organes périphériques qui peuvent leur être adjoints. Les descriptions marquées d'un * se rapportent à des fonctions que ne possèdent que les LOCI spéciales. Les conditions de validité des autres fonctions sont chacunes décrites.

3.1. Touches d'usage général

[0] [1] ..... [9]

Ces touches définissent et visualisent les chiffres successifs dont on forme le nombre dans W (du 2ième au 11ème tube nixie), le tube correspondant au chiffre qui va être placé est éteint et les suivants marquent 0. Si on exécute une opération concernant W sans avoir affiché toutes les positions, les chiffres non affichés manuellement sont pris comme 0, y compris le tube éteint, et la virgule, si elle a été omise, est située par la machine entre le dernier chiffre affiché et le tube éteint.

+ 1 2 3.4 5 6 7 8 9

Système de réponse visuelle qui représente en permanence le contenu du registre W à l'aide de 11 tubes nixie et de 10 petites lampes. Le premier tube représente le signe, les 10 autres les chiffres décimaux qui constituent W, une des 10 petites lampes permet de marquer la position de la virgule.

[+/−]

Cette touche change le signe de W

[.]

Cette touche place la virgule du nombre que l'on entre dans W (si elle est omise, la machine la situe à la droite du dernier chiffre affiché manuellement)

* [←]

Cette touche déplace d'une position à gauche la classe décimale de W qui est prête à être définie manuellement (tube éteint et considéré comme 0 par la machine et précédé de la virgule si celle-ci n'a pas été placée manuellement ailleurs).

* [.←]

Cette touche déplace la virgule d'une position à gauche (W/10→W)

* [.→]

Cette touche déplace la virgule d'une position à droite (W×10→W)

Ces 3 dispositifs ne figurent pas sur les modèles prévus pour être connectés au télétype.

[CL W] Cette touche met W à zéro et MSC à zéro

[CL A] Cette touche met A à zéro

[PRIME] Cette touche met le LOCI en état d'accepter un nouveau travail en mettant tous les registres et MSC à zéro

[CL ER] Cette touche éteint (ou allume) la lampe indicatrice d'erreur.

ERROR  Cette lampe s'allume lors des tentatives d'exécution d'opérations impossible ou lors d'un dépassement de capacité. (ou involontairement par l'instruction 01 (CL ER)

RESPONSE  Cette lampe est allumée pendant que la machine calcule seule. Durant ce laps de temps (imperceptible lorsque l'opérateur travaille manuellement), aucune nouvelle fonction ne peut entrer dans la machine qui est bloquée.

[+] Cette touche provoque l'addition algébrique de W à A sans modifier W
Symboliquement : $A + W \Rightarrow A$

[−] Comme l'addition, soit symboliquement : $A - W \Rightarrow A$

[×] Cette touche provoque l'addition de Ln (W) à L et laisse W à zéro
Symboliquement : $L + Ln\ (W) \Rightarrow L$ et $0 \Rightarrow W$

[÷] Touche de division, elle agit de la même manière que celle de la multiplication, soit
symboliquement : $L - Ln\ (W) \Rightarrow L$ et $0 \Rightarrow W$

[Ln^{-1}] Cette touche place en W l'antilogarithme de L et remet L à zéro
soit symboliquement : $e^{L} \Rightarrow W$ et $0 \Rightarrow L$

[□] Selon la même logique que la multiplication
Symboliquement : $L + 2\ Ln(W) \Rightarrow L$ et $0 \Rightarrow W$

[1/□] Selon la même logique que la division
Symboliquement : $L - 2\ Ln(W) \Rightarrow L$ et $0 \Rightarrow W$

[√] Selon la même logique que la multiplication
Symboliquement : $L + \frac{1}{2}\ Ln(W) \Rightarrow L$ et $0 \Rightarrow W$

[1/√] Selon la même logique que la division
Symboliquement : $L - \frac{1}{2}\ Ln(W) \Rightarrow L$ et $0 \Rightarrow W$

[W → A] Cette touche provoque le transfert de W dans A sans modifier W

[A→W] Cette touche provoque le transfert de A dans W sans modifier A

[W→L] Cette touche provoque le transfert de W dans L sans modifier W, mais y place le point décimal après la 2e position décimale

[L→W] Cette touche provoque le transfert de L dans W et met L à zéro. Les deux premières positions décimales de W étant occupées par la caractéristique, les deux dernières décimales de L sont perdues dans le transfert.

3.2. Touches et voyants pour les mémoires

Les LOCI II peuvent comporter 2, 4 ou 16 mémoires.

[A→S_o] Cette touche provoque le transfert de A dans S_o sans modifier A

[S_o→A] Cette touche provoque le transfert de S_o dans A sans modifier S_o

[W→S_1] [W→S_2] [W→S_3] Ces trois touches provoquent les transferts indiqués sans modifier W

[S_1→W] [S_2→W] [S_3→W] Ces trois touches provoquent les transferts indiqués sans modifier les mémoires S_1 S_2 S_3

[MS] Cette touche provoque, comme l'instruction STEP M, l'avance du contenu du compteur MSC à sa valeur suivante, soit, selon sa logique cyclique :

4 ⇒ MSC si MSC contenait 0
8 ⇒ MSC si MSC contenait 4
12 ⇒ MSC si MSC contenait 8
0 ⇒ MSC si MSC contenait 12

8 ○
4 ○

Ces deux petits voyants lumineux indiquent le contenu du compteur MSC. Ce compteur contient la somme des chiffres dont le voyant est allumé soit :

8 ○	8 ●	8 ☼	8 ☼
4 ○	4 ☼	4 ○	4 ☼
MSC = 0	MSC = 4	MSC = 8	MSC = 12

3.3. Touches, voyants et commutateurs pour la commande du programme

[P_o] Cette touche provoque l'exécution du programme enregistré sur carte à partir de l'instruction No. 0

[P_1] Cette touche provoque l'exécution du programme à partir de l'instruction No. 3

[P_2] Cette touche provoque l'exécution du programme à partir de l'instruction No. 6

Cette touche provoque l'exécution du programme à partir de l'instruction No. 9

Cette touche provoque la continuation du programme à partir d'uneinstruction STOP.

AUTO DISP

Ce petit commutateur, lorsqu'il est placé vers le bas, provoque automatiquement après les additions et les soustractions, le transfert de A dans W sans modifier A, afin de rendre le résultat immédiatement visible.
Dans sa position supérieure, il laisse le contenu de W tel qu'il était et c'est ce dernier qui est affiché sur les tubes nixie.

Note : Lors du fonctionnement programmé (et sauf spécification contraire), il est indispensable de placer le levier vers le haut, faute de quoi des erreurs peuvent intervenir.

STEP
MANUAL
AUTO

Ce petit sélecteur a trois positions :

- dans la position AUTO, la machine exécute à la file les instructions perforées dans les 160 positions des cartes placées dans les lecteurs statiques
- dans la position STEP, la machine n'exécutera qu'une instruction à la fois, chaque fois qu'on appuie sur la touche RUN
- dans la position MANUAL, la machine exécute l'instruction correspondant au numéro de code choisi à l'aide des 6 commutateurs.

3.4. Touches pour commande d'équipements périphériques

Cette touche correspond normalement au retour du chariot (changement de ligne) sur la machine à écrire du Télétype.

Cet ordre peut correspondre à dlautres opérations de contrôle sur des dispositifs d'entrée-sortie spéciaux.

READ

Cette touche est utilisée pour recevoir des données à partir d'un organe d'entrée-sortie (télétype), selon la logique suivante :

a) l'ordre READ a d'abord pour effet de transmettre le contrôle à l'organe de lecture et de bloquer l'avance du registre PC

b) l'organe de lecture transmet l'ordre lu au LOCI

c) le LOCI exécute cet ordre de la façon suivante :

Si l'ordre est RUN, P_0, P_1, P_2, P_3, le contrôle est rendu au lecteur statique selon la logique habituelle. S'il s'agit d'un quelconque autre ordre, il sera exécuté de façon normale, puis automatiquement l'ordre suivant est lu et exécuté, ainsi de suite jusqu'à un STOP, un PRIME ou l'un des ordres ci-dessus rendant le contrôle au lecteur statique.

4) DESCRIPTION DES REGISTRES ET MEMOIRES

4.1. Registres pour toutes les calculatrices

W

Ce registre est celui dans lequel se forme un nombre lorsqu'on le décrit au moyen du clavier ou par la programmation correspondante. C'est lui qui est visible sur les tubes nixie. Il est l'argument de toutes les fonctions arithmétiques et d'entrée-sortie.

A

C'est l'accumulateur arithmétique. C'est donc à lui que s'additionne ou se soustrait W. Il peut être rendu visible par un transfert automatique dans W après toute addition ou soustraction si le commutateur AUTO DISP est baissé.

L

C'est l'accumulateur des logarithmes lors des opérations de multiplication et de division, ainsi que des 4 sous-programmes automatiques.
On peut faire apparaître indifféremment son contenu dans W par le transfert $L \rightarrow W$, ou le résultat dont il est le logarithme par l'ordre Ln^{-1}

4.2. Mémoires pour calculatrices avec mémoires

S_0

C'est une mémoire liée à A par les ordres de transfert $A \rightarrow S_0$, $S_0 \rightarrow A$. C'est également le symbole des mémoires additionnelles S_4, S_8, S_{12} selon l'état du registre MSC.

S_1 S_2 S_3

Ce sont des mémoires liées à W par les 6 ordres de transfert correspondants. Ce sont également les symboles des mémoires additionnelles :

S_5	S_9	S_{13}
S_6	S_{10}	S_{14}
S_7	S_{11}	S_{15}

selon l'état du compteur MSC.

MSC

Ce compteur, sur les modèles à plus de 4 mémoires, peut prendre les valeurs 0, 4, 8, 12. CLW et PRIME le mettent à zéro, l'ordre MS (code STEP M) le fait passer à la position suivante selon une logique cyclique ; il a pour effet de changer la signification des ordres relatifs aux mémoires S_0, S_1, S_2, S_3, en leur affectant les autres mémoires dont l'indice est en réalité :

MSC	pour S_0
MSC + 1	pour S_1
MSC + 2	pour S_2
MSC + 3	pour S_3

4.3. Compteur d'ordre pour LOCI II

PC

Ce registre contient en permanence le numéro d'ordre de l'instruction à exécuter lors d'un calcul contrôlé par un programme sur carte dans le lecteur statique. Son contenu, en représentation décimale, varie donc entre 00 et 79 y compris.

P_0 le met à 00
P_1 le met à 03
P_2 le met à 06
P_3 le met à 09

Il augmente de 1 automatiquement après exécution de toute instruction sauf celles dont la nature est d'agir sur lui, comme par exemple : saut inconditionnel (ordre W→PC). Cet ordre place dans PC les deux premiers chiffres de W et impose ainsi la position de la prochaine instruction à exécuter. D'autres instructions peuvent imposer une valeur calculée à PC (voir les tests, les sous-programmes et leur retour).

X

(pour LOCI II avec deux lecteurs de cartes)
C'est un bit supplémentaire associé à PC et destiné à choisir dans lequel des deux lecteurs statiques le niveau PC détermine la position de la prochaine instruction à exécuter. Un ordre W→PC s'adresse au premier lecteur si W est positif. Il s'adresse au second lecteur si W est négatif.
L'ordre W→X s'adresse forcément au second lecteur de la même façon que W→PC, mais sans tenir compte du signe de W.

DC

Ce compteur contient un nombre de deux chiffres décimaux qui peuvent être placés au moyen de l'ordre W→DC. L'ordre de décrémentation DEC lui soustrait 1. Il est testable par l'ordre DC = 0 ?

PCS	Ce registre conserve le contenu de PC lors du branchement à un sous-programme
DCS	Ce registre conserve le contenu de DC lors du branchement à un sous-programme

5) ENTRETIEN

Le calculateur ne nécessite en principe aucun entretien autre que le nettoyage du lecteur de carte.

En cas de doute sur le bon fonctionnement de l'appareil, faire les tests avec les 8 cartes prévues à cet effet. Répéter les essais et définir avec exactitude l'opération relevée défectueuse en vue du dépannage.

Nettoyage du lecteur de carte

Il faut procéder de temps en temps au nettoyage du lecteur de carte.
Il suffit de frotter doucement la partie fixe avec un chiffon doux ne laissant pas de débris, imbibé de trichloréthylène.
Faire extrêmement attention aux ressorts de contact situés sur le couvercle mobile.
Pour le couvercle, se contenter d'enlever la poussière à l'aide d'un pinceau fin.

6) CONSEILS POUR LA PREPARATION D'UN PROGRAMME

On ne peut rédiger un programme si l'on n'a pas d'abord préparé un schéma de calcul complet dans lequel on aura su expliciter clairement toutes les opérations élémentaires nécessaires à la résolution du problème proposé. On a intérêt à utiliser les feuilles de programmation et à conserver celles-ci avec les cartes programme.

6.1. Le principe essentiel à respecter pour l'élaboration d'un tel schéma est d'une évidence éclatante. Il suffit pourtant d'un seul manque à son application pour aboutir à un échec.
Toute opération ne peut faire appel qu'à des quantités définies. Une quantité étant définie si elle a été antérieurement calculée ou lue (la lecture pouvant être le simple affichage manuel au clavier, une lecture de ruban perforé ou de tout autre organe équivalent). On prendra donc tout d'abord soin de dresser la liste des données du problème, c.a.d. des quantités nécessaires et suffisantes à la détermination des résultats.

6.2. Le second principe est celui de la protection des résultats
Avant de prévoir une opération dont l'effet doit (ou peut) modifier le contenu d'un (ou plusieurs) registres, s'assurer qu'aucun de ces contenus ne doive servir de référence (dans son état actuel) à l'exécution d'une opération ultérieure.

6.3. Le troisième principe est le complémentaire du précédent. Son application n'est indispensable que dans la mesure où l'on risque de ne plus avoir assez de registres de mémoire à sa disposition, c'est donc le principe d'économie des mémoires. Réutiliser dans la mesure du possible, c.a.d. sans négliger le second principe, des mémoires dont le contenu actuel ne doit plus servir.

6.4. Le quatrième principe concerne l'arithmétique : prévoir le comportement arithmétique des variables de façon à éviter (ou au moins à détecter pour en tenir compte) les divisions par 0, les racines carrées d'argument négatif ou toutes autres opérations générant des nombres imaginaires complexes ou simplement hors des limites de représentation de la machine.

Les trois derniers principes concernent la logique proprement dite du programme de calcul.

6.5. Cinquième principe ou principe de sortie des boucles : lorsqu'au moyen d'un branchement en arrière, ou même d'un ensemble de renvois inconditionnels ou sur test, on génère ce qui est appelé une boucle de calcul, ou même un système de boucles imbriquées, il ne faut jamais manquer de s'assurer que le programme, après avoir parcouru le circuit un certain nombre de fois, en sorte pour aboutir à une fin où il donnera ses résultats. Il peut toutefois y avoir une exception pour la boucle la plus extérieure si une suite indéfinie de résultats doit sortir jusqu'à un arrêt manuel.
D'autre part, on boucle généralement un programme sur lui-même de façon à ce qu'il réclame toujours des nouvelles données, après la résolution du cas précédent.

6.6. Sixième principe ou principe d'économie des sauts : lorsqu'un saut sur test définit deux suites logiques, utiliser la séquence naturelle, même si elle contraint à s'occuper d'abord d'une branche du schéma qui se place, en fait, plus loin dans la chronologie des opérations.

6.7. Septième principe ou principe d'économie des opérations : s'efforcer, tout en respectant les autres principes, de ne pas écrire plusieurs fois une même suite d'opérations. Il peut arriver, en effet, que l'on ait prévu des opérations identiques dans des branches parallèles, alors que la logique du problème aurait permis que ces opération aient lieu avant la séparation des branches.
Il peut aussi arriver qu'une suite d'opérations se répète un certain nombre de fois d'une façon invariable et qu'on puisse boucler cette suite dans un comptage. Enfin, on peut faire un sous-programme d'une suite d'opérations qui apparaît en diverses positions du schéma de calcul.

Remarque au sujet des boucles

S'assurer qu'une opération que l'on a placée dans une boucle et dont aucun des arguments ne varie dans cette dernière ne peut être placée avant ou après elle. Ceci vise à améliorer le temps de calcul. S'assurer également de l'application du premier principe en ce qui concerne l'initialisation des grandeurs destinées à varier dans la boucle, en particulier, des sommes cumulées et des indices.

DESCRIPTION DES INSTRUCTIONS PROGRAMMEES

PRINCIPE

Un programme de calcul peut être composé et enregistré sur une ou deux cartes-programme de 80 instructions dont la séquence est guidée par le registre PC, ou sur bande perforée, la capacité étant alors illimitée, mais la logique ne pouvant plus être que séquencielle, les instructions étant exécutées dans l'ordre où elles sont perforées.

Le contrôle pouvant passer automatiquement d'un organe à l'autre, il est aisé de réaliser de vastes programmes enregistrés sur ces trois organes simultanément en utilisant judicieusement leurs qualités propres.

CODIFICATION

Chaque instruction est représentée par un nombre octal de deux chiffres. (C'est-à-dire que seuls les chiffres 0 1 2 3 4 5 6 7 sont utilisés).

Le code de représentation des instructions sur ruban perforé est un code à huit canaux* sa valeur numérique binaire est supérieure de 160 (décimal) à la représentation octale correspondante.

Code octal LOCI	00	à	77
Représentation binaire sur ruban	10100000	à	11011111
Valeur décimale de l'octal	00	à	63
Valeur décimale du binaire	160	-	223

DISPOSITION DU CODE OCTAL SUR CARTE STANDARD IBM DESTINEE AU LECTEUR STATIQUE LOCI

Une instruction occupe une demi colonne, soit 6 positions binaires. On y représente en binaire la valeur octale indiquée. Mais ceci est infiniment plus simple que cette théorie peut laisser supposer, car il suffit de raisonner comme si l'on avait affaire à des nombres décimaux, la somme des positions percées devant être le code à représenter.

PERFORATION

Tenir la carte pré-perforée verticalement devant soi (de façon à ce que les caractères qui y sont imprimés se trouvent dans le bon sens, le coin coupé est alors en haut à droite). Le programme se déroule de haut en bas et de gauche à droite. Leurs numéros d'ordre sont d'ailleurs imprimés entre les bits 1 et 2 pour éviter toute ambiguïté. Ce sont exactement les valeurs de PC correspondantes.

* dont seuls les canaux 1, 2, 3, 4, 5 et 7 sont significatifs pour la LOCI

Pour placer la carte-programme perforée dans le lecteur statique, on doit avoir le coin coupé en haut à gauche. Le programme se déroule alors dans le lecteur de gauche à droite et de bas en haut. Placer la carte soigneusement et refermer à fond.

Note : Les cartes peuvent être perforées très simplement avec la pointe d'un crayon, mais on peut utiliser le Port-à-Punch de IBM.

D'autre part, en cas d'erreur, les trous peuvent être rebouchés recto et verso au moyen de petits morceaux découpés dans du ruban adhésif, mais là aussi il existe des petits carrés adhésifs rouges déjà découpés disposés sur des rouleaux de papier fournis par IBM.

Il est enfin bon de signaler que les trous faits par un perforateur conventionnel dans une carte non pré-emboutie sont parfaitement lisibles par le lecteur statique LOCI.

DESCRIPTION DES CODES OPERATOIRES

00	ignoré par la LOCI		
01	comme la touche	CL ER	
02	comme la touche	CL W	
03	comme la touche	CL A	
04	comme la touche	$\sqrt{\ }$	
05	comme la touche	$1/r$	
06	comme la touche	$\square$	
07	comme la touche	$1/\square$	
10	(step M) comme la touche (fait passer MSC à sa valeur suivante)	MS	
11	comme la touche	WRITE	
12	comme la touche	X	(multiplication)
13	comme la touche	+	(addition)
14	comme la touche	LN^{-1}	
15	comme la touche	-	(soustraction)
16	comme la touche	.	(point décimal)
17	comme la touche	$\frac{\cdot}{\cdot}$	(division)
20	comme la touche	0	
21	comme la touche	1	
22	comme la touche	2	
23	comme la touche	3	
24	comme la touche	4	
25	comme la touche	5	
26	comme la touche	6	
27	comme la touche	7	
30	comme la touche	8	
31	comme la touche	9	
32	comme la touche	RUN	(avancer)
33	comme la touche	$\pm$	(changer le signe de W)
34	Vers. multiplex d'ENTREE (IMX)	(utilisés avec l'option ("d" pour commander des (organes extérieurs à (la LOCI	
35	Vers. Multiplex de SORTIE (OMX)		

36 comme la touche PRIME

37 STOP - C'est l'instruction d'arrêt ; pour passer le contrôle à l'instruction suivante, il faut donner normalement un ordre RUN.

On peut évidemment, selon la nature du programme, continuer par READ, P_0, P_1, P_2, P_3. Cet ordre a trois utilités :

- d'abord en immobilisant le contenu de W, il rend lisible un résultat sur les tubes NIXIE
- ensuite, il permet d'entrer une donnée dans W par le clavier du LOCI. Cette donnée pouvant fort bien être l'issue d'un calcul manuel sur le LOCI, dans la mesure où ce calcul ne recouvre par inopinément des registres utiles à la continuation du programme
- enfin, il laisse l'opérateur intervenir en choisissant la branche du programme à laquelle il doit rendre le contrôle.

40 W⟶PC

Cet ordre romp la séquence naturelle du registre PC par laquelle les instructions s'exécutent l'une après l'autre, en plaçant dans PC les deux premiers chiffres de W, imposant ainsi le numéro de la prochaine instruction à exécuter. Si W ne contient qu'un chiffre, le second est pris pour un zéro (ainsi [4], [W⟶PC] met PC à 40; [0], [4], [W⟶PC] met PC à 4

Si W est négatif, la prochaine instruction dont le numéro est imposé l'est dans le second lecteur statique, sinon dans le premier.

41 W⟶XPC

Cet ordre agit comme le précédent (W⟶PC), mais s'adresse toujours au second lecteur, indépendamment du signe de W.

42 W⟶DC

Cet ordre place dans DC les deux premiers chiffres de W, mais ici [4], [W⟶DC] met DC à 4, pour le mettre à 40 il faut [4] [0], [W] [DC]

43 DC⟶W

Cet ordre place les deux digits de DC dans W, suivis du point décimal et de huit zéros.

44	comme la touche	W → A
45	comme la touche	A → W
46	comme la touche	W → L
47	comme la touche	L → W
50	comme la touche	A → S_o
51	comme la touche	S_o → A
52	comme la touche	W → S_1
53	comme la touche	S_1 → W
54	comme la touche	W → S_2
55	comme la touche	S_2 → W
56	comme la touche	W → S_3
57	comme la touche	S_3 → W
60	P_o	(exécuter en 00 du premier lecteur)
61	P_1	(exécuter en 03 du premier lecteur)
62	P_2	(exécuter en 06 du premier lecteur)
63	P_3	(exécuter en 09 du premier lecteur)
64	APPEL	(c'est l'ordre qui envoie dans un sous-programme)

Cet ordre se comporte comme le saut inconditionnel (W → PC) mais conserve les contenus de DC et de PC dans les registres auxiliaires DCS et PCS. Le premier avantage est de pouvoir faire usage du registre DC dans les divers sous-programmes et dans le programme principal de façon indépendante, c'est-à-dire comme s'il s'agissait chaque fois d'un autre registre. D'autre part, c'est une nécessité de pouvoir continuer le programme principal dans sa séquence naturelle lors du retour du sous-programme. C'est pourquoi l'adresse de retour (celle qui suit l'appel) est conservée dans PCS.

Voici en résumé, la succession des opérations générées par la rencontre du code 64 :

1) PC → PCS (adresse de retour)
2) DC → DCS (protection du compteur)
3) W → PC (définition de l'adresse du sous-programme)

65 C'est l'ordre de retour d'un sous-programme au programme principal

Cet ordre restaure le compteur DC tel qu'il était lors de l'appel et donne le contrôle à l'instruction qui suit l'appel.

En résumé, voici la succession des opérations générées par la rencontre du code 65 :

1) DCS → DC (restauration du compteur)
2) PCS → PC (définition de l'adresse de retour)
3) PC → PCS (protection de l'adresse de fin du sous-programme)

36 comme la touche PRIME

37 STOP - C'est l'instruction d'arrêt ; pour passer le contrôle à l'instruction suivante, il faut donner normalement un ordre RUN.

On peut évidemment, selon la nature du programme, continuer par READ, P_0, P_1, P_2, P_3. Cet ordre a trois utilités :

- d'abord en immobilisant le contenu de W, il rend lisible un résultat sur les tubes NIXIE
- ensuite, il permet d'entrer une donnée dans W par le clavier du LOCI. Cette donnée pouvant fort bien être l'issue d'un calcul manuel sur le LOCI, dans la mesure où ce calcul ne recouvre par inopinément des registres utiles à la continuation du programme
- enfin, il laisse l'opérateur intervenir en choisissant la branche du programme à laquelle il doit rendre le contrôle.

40 W⟶PC

Cet ordre romp la séquence naturelle du registre PC par laquelle les instructions s'exécutent l'une après l'autre, en plaçant dans PC les deux premiers chiffres de W, imposant ainsi le numéro de la prochaine instruction à exécuter. Si W ne contient qu'un chiffre, le second est pris pour un zéro (ainsi [4], [W⟶PC] met PC à 40; [0], [4], [W⟶PC] met PC à 4

Si W est négatif, la prochaine instruction dont le numéro est imposé l'est dans le second lecteur statique, sinon dans le premier.

41 W⟶XPC

Cet ordre agit comme le précédent (W⟶PC), mais s'adresse toujours au second lecteur, indépendamment du signe de W.

42 W⟶DC

Cet ordre place dans DC les deux premiers chiffres de W, mais ici [4], [W⟶DC] met DC à 4, pour le mettre à 40 il faut [4] [0], [W] [DC]

43 DC⟶W

Cet ordre place les deux digits de DC dans W, suivis du point décimal et de huit zéros.

Remarque au sujet des sous-programmes

Les ordres d'appel et de retour doivent alterner, ce qui signifie qu'un sous-programme ne peut en appeler un autre. On peut rencontrer plusieurs RETOUR successifs, ayant pour effet de renvoyer à l'ordre suivant le précédent RETOUR, mais en reprenant à chaque fois la valeur initiale de DC.

En revanche, l'entrée d'un sous-programme n'est pas une instruction singulière, de sorte qu'un sous-programme peut avoir plusieurs entrées représentant des variantes n'ayant qu'une certaine partie commune.

Enfin, si un sous-programme est constitué de plusieurs branches terminales, chacune peut posséder son ordre de retour.

Tous les registres en virgule flottante (sauf W) peuvent être considérés comme variables d'entrée d'un sous-programme, tous comme variables de sortie. En bref, ce couple d'instructions permet tous les arrangements, sauf l'imbrication.

66 C'est l'ordre de décomptage. Il soustrait 1 au compteur DC

67 Test d'erreur

Si le signal d'erreur n'est pas allumé, le contrôle est donné à l'instruction qui se trouve quatre positions plus loin dans la carte, sautant donc en séquence trois positions. Si le signal erreur est au contraire allumé, le programme continue en séquence, il rencontre donc les trois positions laissées par l'autre branche, et ces dernières peuvent servir à faire sauter ailleurs (deux digits à charger dans W, puis W → PC). En conjonction avec l'ordre 01 (CL ER), permet d'utiliser la bascule ERREUR comme une mémoire supplémentaire de capacité 1 bit.

70 Test DC = 0 ?

Si le registre DC contient zéro, le contrôle est donné à l'instruction qui se trouve quatre positions plus loin dans la carte, sautant donc en séquence trois positions. Si, au contraire, DC est différent de zéro, le programme continue en séquence, il rencontre donc les trois positions laissées par l'autre branche, et ces dernières peuvent servir à faire sauter ailleurs (deux digits à charger dans W, puis W → PC).

71 Test A = 0 ?

Même logique que le code précédent (DC = 0 ?) mais appliquée à l'accumulateur A.

72 Test W = 0 ?

Même logique que les deux codes précédents (DC = 0 ? et A = 0 ?) mais appliquée qu registre central W. Dans les LOCI équipées du CU-1 ou CU-2 (boîte de commande d'un lecteur de bande perforée) cet ordre signifie LIRE cette bande.

73 Test $W < 0$?

Selon la même logique que pour les quatre codes précédents, le contrôle passe à l'instruction qui se trouve quatre positions plus loin dans la carte si le contenu de W est négatif, s'il est positif le programme continue en séquence.

74 Test $L < 0$?

Selon la même logique que tous les autres codes de test qui précèdent, le programme saute ou ne saute pas trois instructions, mais il est important de souligner qu'il s'agit du registre logarithmique et que, relativement au résultat dont il contient le logarithme naturel (appelons abstraitement ce résultat R), ce test est à interpréter pratiquement comme : $0 < R < 1$? et sert essentiellement à tester les ordres de grandeur.
Attention ! Ce test ne remet pas L à zéro. Ne pas oublier de le faire (par PRIME, LN^{-1} ou $L \rightarrow W$) si l'on désire ensuite recommencer de nouveaux calculs.

75 comme la touche CR

76 comme la touche READ

77 ignoré par la LOCI (ce qui permet l'annulation d'une erreur de perforation)

ANALOGIES ENTRE LE CODE LOCI ET LE LANGAGE INTERPRETATIF "FORTRAN"

ORDRES DU LOCI	SYMBOLES "FORTRAN"
Arithmétique	
+	+
-	-
X	*
÷	/
±	W = -W
L → W	LØGF (ce qui a servi à former le contenu de L)
LN^{-1}	EXPF (contenu véritable de L)
√‾	SQRTF (W)
□	W**2
$1/\sqrt{\ }$	/SQRTF (W)
1/□	/W**2
Clear W) Clear A)	(W = O (A = O
Logique	
	GØ TØ ... ou GØ TØ (...), ...
W → PC W → XPC	(FUNCTION (appel) (SUBROUTINE (call)
Store et Jump	RETURN
Restore	
W → DC) DEC) DC = O ?) W → PC)	D Ø ... ...
A = O ?) W = O ?) W < O ?) L < O ?) W → PC)	IF (...) ..., ..., ...
P_o P_1 P_2 P_3	IF (SENSE SWITCH ...) ..., ...
STEP M	(Peut permettre de simuler (l'indicage simple S(M) au (sens Sm

Entrée-sortie

STOP	ACCEPT ...
	PAUSE
READ	ACCEPT TAPE ...
WRITE)	(PRINT ...
CR)	(TYPE ...
	(PUNCH TAPE ...

Exemple I - RESOLUTION DES EQUATIONS DU SECOND DEGRE AVEC LEURS DEUX RACINES SOUS FORME DE NOMBRES COMPLEXES

Ce programme fournit des exemples sur les sujets suivants :

1) Arithmétique (non transcendante)
2) Utilisation des mémoires de première génération (c'est-à-dire sans agir sur le registre MSC)
3) Saut inconditionnel
4) Test et saut sur test

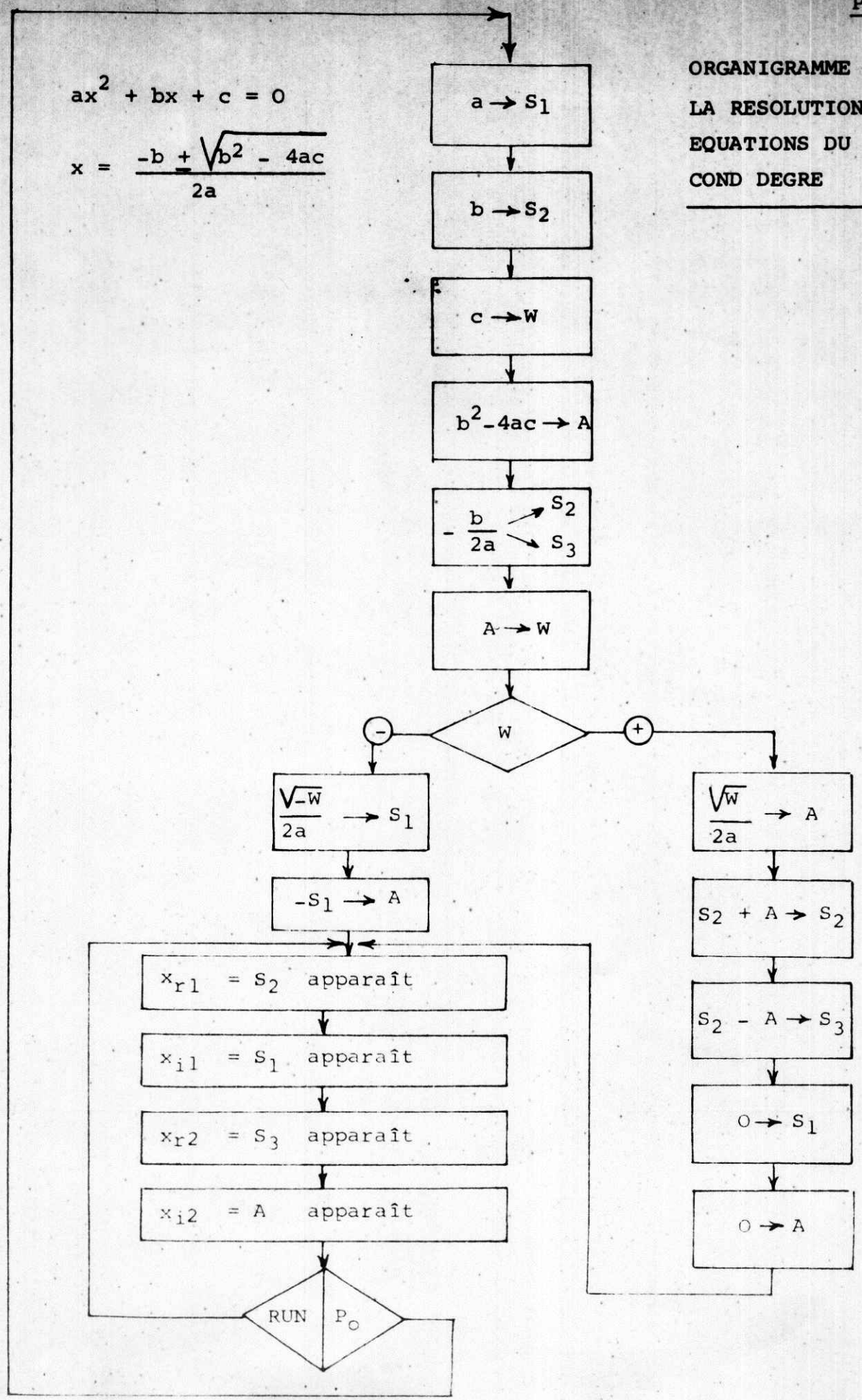
On y rencontre les codes suivants :

04	$\sqrt{\ }$	40	W → PC
06	$\square$	44	W → A
12	$\times$	45	A → W
13	$+$	52	W → S_1
14	LN^{-1}	53	S_1 → W
15	$-$	54	W → S_2
17	$\div$	55	S_2 → W
de 20 à 31	Des chiffres entrant dans W	56	W → S_3
33	$\pm$	57	S_3 → W
36	PRIME	73	W < 0 ?
37	STOP		

ORGANIGRAMME POUR LA RESOLUTION DES EQUATIONS DU SECOND DEGRE

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



Programme L O C I

Equation du second degré :

$ax^2 + bx + c = 0$

Solution : $\begin{cases} x_{r1} + ix_{i1} \\ x_{r2} + ix_{i2} \end{cases}$

F o n c t i o n n e m e n t :

1) [Po]
2) Donner à
3) [RUN]
4) Donner b
5) [RUN]
6) Donner c
7) [RUN]
8) x_{r1} apparaît
9) [RUN]
10) x_{i1} apparaît
11) [RUN]
12) x_{r2} apparaît
13) [RUN]
14) x_{i2} apparaît
15) [RUN] → 8)

No	Cmd	Code	Comment.	No	Cmd	Code	Comment.	No	Cmd	Code	Comment.	No	Cmd	Code	Comment.
00	PRIME	36		20	2	22		40	$W \to S_1$	52	$x_{i1} \to S_1$	60	Ln^{-1}	14	$\sqrt{D}/(2a) \to W$
01	STOP	37		21	÷	17		41	±	33		61	$W \to A$	44	$\sqrt{D}/(2a) \to A$
02	$W \to S_1$	52	$a \to S_1$	22	$S_1 \to W$	53	$a \to W$	42	$W \to A$	44	$(x_{i2} = x_{i1}) \to A$	62	$S_2 \to W$	55	$-b/(2a) \to W$
03	STOP	37		23	÷	17		43	$S_2 \to W$	55		63	+	13	x_{r1} final $\to A$
04	$W \to S_2$	54	$b \to S_2$	24	Ln^{-1}	14	$b/(2a) \to W$	44	STOP	37	x_{r1} apparaît	64	$A \to W$	45	x_{r1} final $\to W$
05	STOP	37	$c \to W$	25	±	33	$-b/(2a) \to W$	45	$S_1 \to W$	53		65	$W \to S_2$	59	x_{r1} final $\to S_2$
06	×	12		26	$W \to S_2$	54	$x_{r1} \to S_2$	46	STOP	37	x_{i1} apparaît	66	$S_2 \to W$	57	$-b/(2a) \to W$
07	$S_1 \to W$	53	$a \to W$	27	$W \to S_3$	56	$(x_{r2} = x_{r1}) \to S_3$	47	$S_3 \to W$	57		67	−	15	
08	×	12		28	$A \to W$	45	$D \to W$	48	STOP	37	x_{i2} apparaît	68	−	15	
09	4	24		29	$W < 0$?	73	$D < 0$?	49	$A \to W$	45		69	$A \to W$	45	$b/(2a) + \sqrt{D}/(2a)$
10	×	12		30	5	25		50	STOP	37	x_{i2} apparaît	70	±	33	
11	Ln^{-1}	14	$4ac \to W$	31	4	24		51	4	24		71	$W \to S_3$	56	x_{r2} final $\to S_3$
12	±	33		32	$W \to PC$	40	là $D > 0$	52	3	23		72	0	20	
13	$W \to A$	44	$-4ac \to A$	33	±	33	ici $D < 0$	53	$W \to PC$	40		73	$W \to S_1$	52	$(x_{i1} = 0) \to S_1$
14	$S_2 \to W$	55	$b \to W$	34	√	04	$\sqrt{-D}$	54	$A \to W$	45	ici $D > 0$	74	$W \to A$	44	$(x_{i2} = 0) \to A$
15	□	06		35	2	22		55	√	04		75	4	24	
16	Ln^{-1}	14	$b^2 \to W$	36	÷	17		56	2	22		76	3	23	
17	+	13	$b^2 - 4ac \to A$	37	$S_1 \to W$	53	$a \to W$	57	÷	17		77	$W \to PC$	40	
18	$S_2 \to W$	55	$b \to W$	38	÷	17		58	$S_1 \to W$	53	$a \to W$	78			
19	×	12		39	Ln^{-1}	14	$\sqrt{-D}/(2a) \to W$	59	÷	17		79			

Exemple II - UN SOUS-PROGRAMME D'INDIÇAGE

Il est fréquent qu'un calcul nécessite la mémorisation d'un certain nombre de valeurs selon un classement leur attribuant une relation d'ordre (la première, la seconde, la troisième,... la Nème).

On veut donc opérer dans un registre de mémoire S_i pour $1 \leq i \leq N$ et i pouvant être l'issue d'un calcul quelconque, étant bien entendu qu'à une valeur de i donnée ne correspond qu'une mémoire bien définie et réciproquement. Il faut également prévoir un système d'indiçage dans les deux sens $W \rightarrow S_i$ et $S_i \rightarrow W$

Le sous-programme dont il s'agit ici fonctionne de la façon suivante :

Il est conçu pour le second lecteur statique seulement. Le programme principal qui est dans le premier lecteur dispose des mémoires S_j pour $1 \leq i \leq 16$ au moyen des séquences suivantes :

Pour mémoriser une quantité Q à un indice i

Conventions : LnQ est dans L
i est dans A

Séquence :
o)
+) soit trois instructions
Store et Jump)

Pour reprendre une quantité Q mémorisée à l'indice i

Convention : i est dans A (Q qui est dans S_i) $\rightarrow$ W

Séquence :
o)
1)
+) soit quatre instructions
Store et Jump)

Les valeurs de i ne coïncident pas avec la classification symbolique des mémoires LOCI (S1 pour $0 \leq 1 \leq 15$).

Pour faciliter la compréhension de ce sous-programme, voici la table de correspondance :

classé selon j

Indice math.	Registre LOCI		
i	l	STEP	indice de co-dage
1	5	1	1
2	6	1	2
3	7	1	3
4	8	2	0
5	9	2	1
6	10	2	2
7	11	2	3
8	12	3	0
9	13	3	1
10	14	3	2
11	15	3	3
12	0	4	0
13	1	4	1
14	2	4	2
15	3	4	3
16	4	5	1

classé selon l

Registre LOCI			Indice math.
l	STEP	indice de co-dage	i
0	0	0	12
1	0	1	13
2	0	2	14
3	0	3	15
4	1	0	16
5	1	1	1
6	1	2	2
7	1	3	3
8	2	0	4
9	2	1	5
10	2	2	6
11	2	3	7
12	3	0	8
13	3	1	9
14	3	2	10
15	3	3	11

Ce sous-programme fournit des exemples sur les sujets suivants :

1) Utilisation de toutes les mémoires (emploi du registre MSC)

2) Saut inconditionnel et sur tests

3) Conception d'un sous-programme

On y rencontre les codes suivants :

02	CLW	52	$W \rightarrow S_1$
10	STEP	53	$S_1 \rightarrow W$
13	+	54	$W \rightarrow S_2$
14	LN^{-1}	55	$S2 \rightarrow W$
15	-	56	$W \rightarrow S_3$
41	$W \rightarrow$ XPC	57	$S_3 \rightarrow W$
42	$W \rightarrow$ DC	65	Restore
44	$W \rightarrow$ A	70	DC = 0 ?
45	$A \rightarrow W$	71	A = 0 ?
50	$A \rightarrow S_0$	73	W < 0 ?
51	$S_0 \rightarrow A$		

Programme L O C I

SOUS PROGRAMME D'INDIÇAGE (A placer dans le second lecteur)

(Les positions 58 à 80 sont disponibles pour un autre sous-programme)

Fonctionnement :

l'indice $1 \leq i \leq 16$ doit être dans le registre A lors de l'appel de ce sous programme

Mémoriser un nombre à l'indice i

Le log de ce nombre doit être dans le registre L lors de l'appel

Appel : 0 + St.& J.

Reprendre un nombre mémorisé à l'indice i

Le nombre mémorisé à l'indice i se trouvera dans le reg.W lors du retour

Appel : 0 1 + St.& J.

L'indice d'appel est détruit par ce sous-programme

No	Cmd	Code	Comment.	No	Cmd	Code	Comment.	No	Cmd	Code	Comment.	No	Cmd	Code	Comment.
00	0			20	W→S3			40	S3→W			60			
01	W→X			21	Rest.			41	Rest.						
02	CLW			22	1			42							
03	4			23	+			43	1						
04	-			24	A→W			44	+						
05	A→W			25	W<0?			45	A→W			65			
06	STEP			26	LN-1			46	W<0?						
07	W<0?			27	W→S2			47	S2→W						
08	0			28	Rest.			48	Rest.						
09	3			29	1			49							
10	W→X			30	+			50	1			70			
11	1			31	LN-1			51	+						
12	+			32	A=0?			52	A=0?						
13	DC=0?			33	W→A			53	S0→A						
14	3			34	A→S0			54	A→W						
15	8			35	Rest.			55	Rest.			75			
16	W→XPC			36	W→S1			56	S1→W						
17	A→W			37	Rest.			57	Rest.						
18	W<0?			38	A→W			58							
19	LN-1			39	W<0?			59							

Exemple III - SOUS-PROGRAMME POUR LE CALCUL DES ARC-TANGENTES

Ce sous-programme n'est également conçu que pour le second lecteur. Il suppose que lors de l'appel le Ln de la tangente est en L (ce qui est naturel lorsqu'il s'agit d'un rapport). Au retour, l'angle en degrés et fraction décimale du degré, se trouve en W.

L'appel, depuis le premier lecteur, se rédige par les trois instructions suivantes :

o
+
Store et Jump

Le système employé est un simple développement en série de Mac-Laurin avec complémentation automatique pour les valeurs absolues supérieures à 1.

Le développement en série n'est interrompu que lorsque les termes deviennent inférieurs à la limite de représentation de la machine ($<10^{-10}$).

Ce sous-programme fournit des exemples sur les sujets suivants :

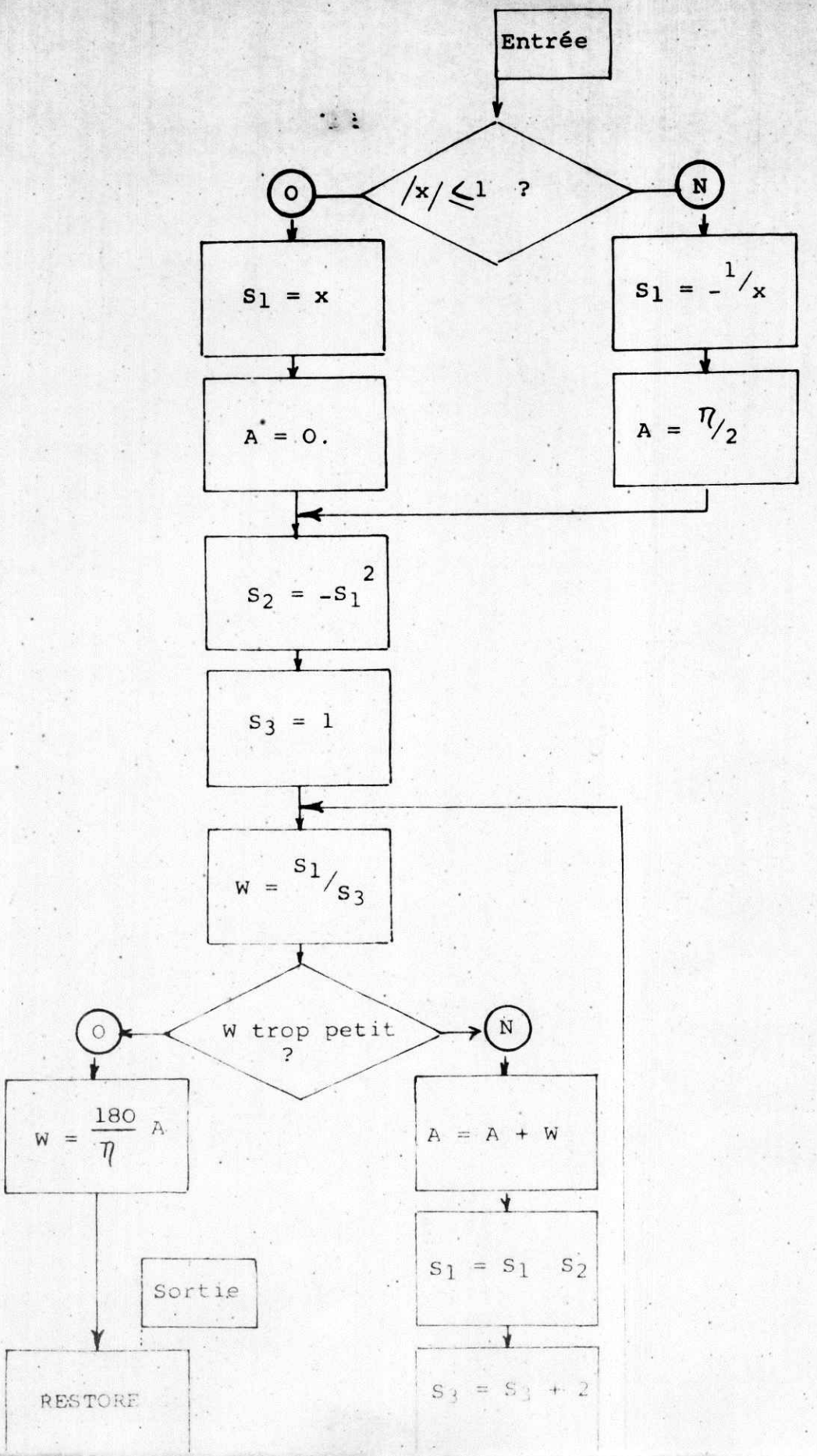
1) Arithmétique (non transcendante)
2) Bouclage et test sur convergence
3) Conception d'un sous-programme

On y rencontre les codes suivants :

01	CL error	47	L → W
06	□	50	A → S_0
12	×	51	S_0 → A
13	+	52	W → S_1
14	LN^{-1}	53	S_1 → W
16	·	54	W → S_2
17	$\frac{\cdot}{\cdot}$	55	S_2 → W
33	$\pm$	56	W → S_3
41	W → XPC	57	S_3 → W
44	W → A	65	Restore
45	A → W	67	Error test ?
46	W → L	74	test L < O ?

ORGANIGRAMME DU DEVELOPPEMENT EN SERIE DE MAC-LAURIN DE

W = arctgx POUR PROGRAMMATION SUR LOCI



Programme L O C I

Arc tg x

Convention :

Lors de l'appel : ln x est en L

Au retour : arctg est en W
(en degrés et décimales)

Séquence d'appel (depuis le premier lecteur)

o
+
Store & Jump

Soit trois instructions

No	Cmd	Code	Comment.	No	Cmd	Code	Comment.	No	Cmd	Code	Comment.	No	Cmd	Code	Comment.
00	L < 0 ?	74	|x| ≤ 1 ?	20	6	26		40	6	26		60	0	20	
01	4	24		21	3	23		41	W → XPC	41	boucle	61	8	30	
02	2	22		22	W → XPC	41		42	L → W	47		62	W → XPC	41	rejoindre la
03	W → XPC	41		23	+	13	A = A + W	43	±	33		63	CL ER	01	même sortie
04	LN⁻¹	14		24	S_1 → W	53		44	W → L	46	$S_1 = -1/x$	64	5	25	
05	W → S_1	52	$S_1 = x$	25	×	12		45	LN⁻¹	14		65	7	27	
06	0	20		26	S_2 → W	55		46	±	33		66	.	16	
07	W → A	44	A = 0	27	×	12		47	W → S_1	52		67	2	22	
08	S_1 → W	53		28	LN⁻¹	14		48	1	21		68	9	31	
09	□	06		29	W → S_1	52	$S_1 = S_1\ S_2$	49	.	16		69	5	25	180/π
10	LN⁻¹	14		30	A → S_0	50	protection	50	5	25		70	7	27	
11	±	33		31	S_3 → W	57		51	7	27		71	7	27	
12	W → S_2	54	$S_2 = -S_1^2$	32	W → A	44		52	0	20		72	9	31	
13	1	21		33	2	22	$S_3 = S_3 + 2$	53	7	27	π/2	73	5	25	
14	W → S_3	56	$S_3 = 1$	34	+	13		54	9	31		74	1	21	
15	—	17		35	A → W	45		55	6	26		75	×	12	
16	S_1 → W	53		36	W → S_3	56		56	3	23		76	A → W	45	
17	×	12		37	S_0 → A	51	restauration	57	2	22		77	×	12	
18	LN⁻¹	14	$W = S_1/S_3$	38	÷	17	$1/S_3$	58	7	27		78	LN⁻¹	14	$W = \frac{180}{\pi} \cdot A$
19	error code	67	terme négligeable ?	39	1	21		59	W → A	44	$A = \pi/2$	79	RESTORE	65	Retour

Exemple IV - QUATRE FONCTIONS TRANSCENDANTES

Voici la description d'une carte-programme destinée à mettre en évidence et à donner un accès facile aux fonctions transcendantes auxquelles on a un accès direct avec le calculateur LOCI.

1) 10^x Ce programme donne l'antilogarithme décimal de x

Donner [Po]
Donner x
Lire le résultat

2) $\text{Log}_{10}x$ Ce programme donne la mantisse du logarithme décimal de x

Donner [P1]
Donner x sans s'occuper du point décimal
(ce dernier est placé par le programme)
Lire la mantisse du log

Remarque : Ces deux premiers programmes remplacent d'énormes tables de log décimaux à longues mantisses. Ces tables sont rares et d'un emploi pénible.

3) b^p Ce programme réalise l'exponentiation de deux variables

Donner [P2]
Donner b
Donner [RUN]
Donner p
Lire b^p l'exponentielle générale

4) $\log_b n$ Ce programme calcule la logarithmique générale de deux variables

Donner [P3]
Donner b (base du log)
Donner [RUN]
Donner n (argument) puis RUN
Lire log de base b de n

Cette carte est un exemple pour les sujets suivants :

1) Arithmétique transcendante
2) Utilisation manuelle des fonctions Po, P_1, P_2, P_3
3) Sauts inconditionnels à des fins de programmes communes
4) Exemple d'un arrêt bouclé sur lui-même afin d'éviter un égarement dans des instructions imprévues. (obligation de donner à chaque calcul le Po, P1, P2 ou P3).