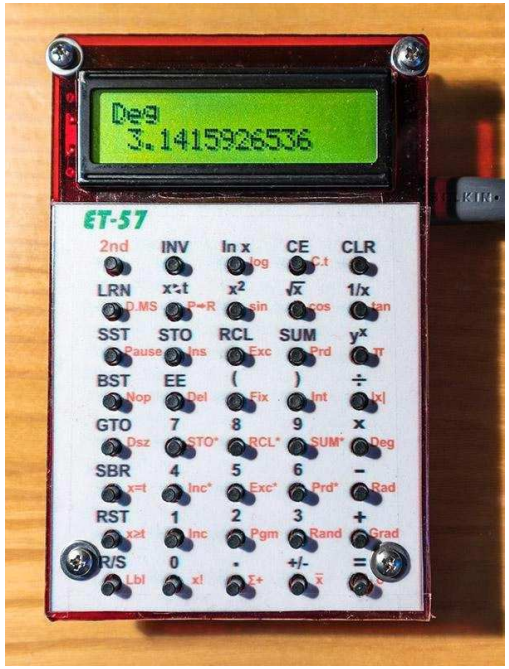
CALCULATRICE PROGRAMMABLE

ET-57



Manuel de l'Utilisateur

Calculatrice programmable ET-57

Manuel de l'Utilisateur

Calculatrice version 201101

Mars 2023

d'après le manuel original de Miroslav Nemecek
traduction et adaptation de Pierre Houbert

site Internet: http://www.breatharian.eu/hw/et57/

Sommaire

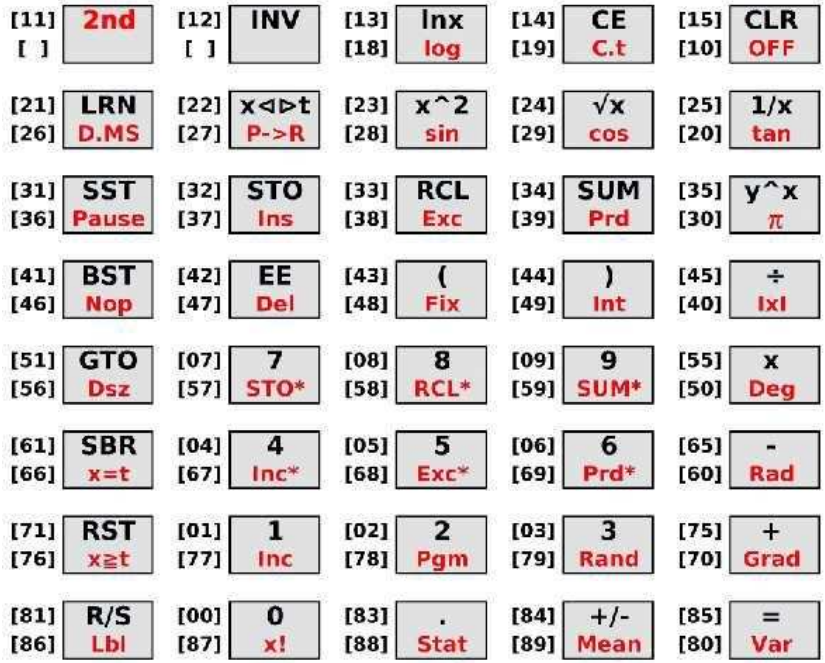
1		Disposition du clavier	6
2		Caractéristiques	8
3		Description	9
4		Comment utiliser la calculatrice	10
5		Écarts par rapport à la TI-57	11
6		Format des nombres	12
7		Clavier	14
8		Indicateurs à l'écran	15
9		Éditeur de nombre	16
10		Expressions numériques	17
11		Adressage des Registres	18
12		La programmation	22
13		Périphériques et ports externes	24
14		Touches et instructions	29
0	**0...9**	Chiffres de base	29
10	**OFF**	Éteindre la calculatrice (ET-57B)	29
11	**2nd**	Fonction alternative	29
12	**INV**	Inverse de fonction	30
13	**lnx**	Logarithme népérien et exposant	30
14	**CE**	Correction d'erreur de saisie	30
15	**CLR**	Effacement de l'affichage	31
18	**log**	Logarithme décimal et exposant	31
19	**C.t**	Effacer le registre T	31
20	**tan**	Tangente	31
21	**LRN**	Programmation	32
22	**x<>t**	Echange des registres X et T	32
23	**x^2**	Carré d'un nombre	32
24	**Vx**	Racine carrée d'un nombre	32
25	**1/x**	Inverse d'un nombre	32
26	**D.MS**	Conversions minutes/secondes	33
27	**P->R**	Conversion polaire/cartésien	34
28	**sin**	Sinus	34
29	**cos**	Cosinus	35
30	**pi**	Nombre de Ludolf	35
31	**SST**	Avance d'un pas dans le programme	35
32	**STO**	Enregistrement dans un registre	35
33	**RCL**	Rappel du contenu d'un registre	35
34	**SUM**	Addition dans un registre	36
35	**y^x**	Puissance et racine	36
36	**Pause**	Délai d’attente	36
37	**Ins**	Insertion d'un pas dans le programme	36
38	**Exc**	Échange X avec un registre	36
39	**Prd**	Multiplier/diviser dans un registre	37
40	**IxI**	Valeur absolue	37
41	**BST**	Recul d’un pas dans le programme	37
42	**EE**	Mode exposant	37
43	**(**	Parenthèse gauche	37
44	**)**	Parenthèse droite	37
45	**:**	Division	38
46	**Nop**	Aucune opération	38
47	**Del**	Suppression du pas courant	39
48	**Fix**	Nombre de décimales	39
49	**Int**	Partie entière	39
50	**Deg**	Degrés	39
51	**GTO**	Saut Saut vers label (ou adresse)	39
55	**x**	Multiplication	39
56	**Dsz**	Boucle de programme	40
57	**STO***	Stockage indirect dans un registre	41
58	**RCL***	Rappel indirect d’un registre	41
59	**SUM***	Addition indirecte dans un registre	41
60	**Rad**	Radians	42
61	**SBR**	Appel sous-programme	42
65	**-**	Soustraction	42
66	**x=t**	Test égalité	42

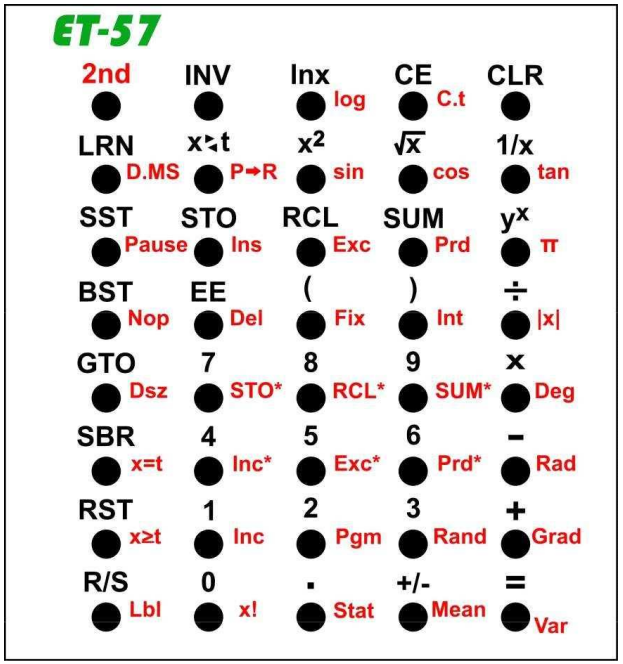
67	**Inc***	Incrément/décrément de registre indirect	42
68	**Exc***	Échange indirect X avec un registre	43
69	**Prd***	Produit indirect dans registre	43
70	**Grd**	Grades	43
71	**RST**	Réinitialisation	43
75	**+**	Ajout	43
76	**x>=t**	Supérieur ou égal à	44
77	**Inc**	Incrémente / décrémente un registre	44
78	**Pgm**	Sélection de l'espace de programme	44
79	**Rand**	Générateur de nombres aléatoires	45
80	**Var**	Dispersion	46
81	**R/S**	Démarrage et arrêt du programme	46
83	**.**	Point décimal	47
84	**+/-**	Changement de signe	47
85	**=**	Effectuer le calcul	47
86	**Lbl**	Labels	47
87	**x!**	Factorielle	48
88	**Stat**	Statistiques	48
89	**Mean**	Moyenne	49

15	Exemples de programmes	50
1	Lancer les dés (Version ET-57)	50
2	Lancez les dés (Version TI-57)	51
3	Serpent lumineux LED	52
4	Calcul du polynôme	53
5	Nombres complexes	55
6	Approximation de Ramanujan de la factorielle x !	61
7	Approximation de Stirling de la factorielle ln(x!)	63
8	Détermination des zéros d'une fonction	66
9	Intégrale de Simpson de la fonction	71
10	Droite de régression linéaire	74

1. Disposition du clavier

Pour chaque touche, la signification de base est indiquée sur la 1ère ligne et la signification alternative sur la 2ème ligne de l'écran.
(après avoir appuyé sur la touche 2nd).





2. Caractéristiques

Résumé : Précision de calcul de 17 chiffres de mantisse (code DCB), 11 chiffres de mantisse affichés. 80 registres (RAM), 500 pas de programme (EEPROM), processeur ARmega8, écran LCD.

- Processeur ATmega8 (8 MHz, 8 Ko de ROM, 1 Ko de RAM, 512 B d'EE-PROM)
- Tension d'alimentation 5 V (à partir d'un chargeur USB ou d'un port USB)
- Calculs en code DCB
- Clavier 40 touches
- Écran LCD à deux lignes (2 x 16 caractères alphanumériques)
- Précision de calcul de 17 chiffres
- Précision des registres de base 15 chiffres
- Précision des registres étendus 13 chiffres
- Affichage des données à 11 chiffres significatifs
- Mode d'affichage scientifique avec exposant sur 2 chiffres, de - 99 à +99
- 10 espaces programme
- 50 pas de programme par espace (500 pas au total)
- 10 labels par espace de programme
- Appels de fonction et sauts entre les espaces de programme
- Programme utilisateur stocké en EEPROM (sans pile)
- 10 registres de base (accessibles par adressage direct)
- 70 registres étendus (accessibles par adressage d'indirect)
- Indexation accès aux variables
- Fonctions exponentielles et logarithmiques
- Fonctions trigonométriques
- Fonctions statistiques
- Factorielle
- Générateur de nombres aléatoires
- Matériel et logiciel entièrement open source
- Contrôle d'un appareil externe via le connecteur ISP
- Code de la calculatrice entièrement écrit en assembleur AVR

3. Description

La calculatrice **ET-57** est conceptuellement basée sur la célèbre calculatrice **TI-57**, développée en 1977 par ***Texas Instruments***.
Elle essaie de maintenir la compatibilité avec les programmes écrits pour la **TI-57** tout en élargissant les fonctionnalités en utilisant les capacités du processeur utilisé.
La fonctionnalité est étendue par plus d'espaces de programme (10 espaces pour un total de 500 pas), plus de registres de données (80), adressage direct et indirect, factoriel, générateur de nombres aléatoires....

La calculatrice est destinée aux anciens utilisateurs de la calculatrice **TI-57**, à ceux qui s'intéressent à la technologie rétro, et comme aide pédagogique pour se familiariser avec les principes d'utilisation et de programmation de la calculatrice.
Pour cette raison, ils s'efforcent de simplifier au maximum la conception, composée uniquement de micro-interrupteurs, d'un processeur, d'un écran LCD, d'un connecteur pour l'alimentation externe et de quelques petits composants.
La **ET-57** est un outil fait pour expérimenter, enseigner, ou être utilisée au bureau avec une alimentation externe à partir d'un chargeur USB ou d'un port USB.

En plus de la variante de base **ET-57** avec un processeur ATmega8, la calculatrice peut aussi être également disponible dans la variante matérielle de la calculatrice **ET-58**, avec les processeurs ATmega88, ATmega168 ou ATmega328.
Cette variante **ET-57B** diffère de la variante de base en ce qu'elle permet d'éteindre la calculatrice (bouton OFF) et de contrôler le contraste de l'écran LCD, mais n'offre pas d'accès aux périphériques externes (la calculatrice **ET-58** n'inclut pas un connecteur ISP).

4. Comment utiliser la calculatrice

La calculatrice **ET-57** est équipée d'un écran LCD alphanumérique à 2 lignes, de 40 micro-interrupteurs et d'un processeur.

Étant donné que la calculatrice n'est pas alimentée par une batterie (elle est destinée à une utilisation de bureau avec une alimentation externe à partir d'un chargeur USB ou d'un port USB), elle ne comprend pas d'interrupteur d'alimentation.
Le programme utilisateur est stocké dans la mémoire EEPROM dont le contenu est conservé même sans alimentation.
En débranchant l'alimentation, la calculatrice est réinitialisée, les registres, le contenu de l'afficheur et les opérations démarrées sont effacés, seul le contenu du programme utilisateur (dans l'EEPROM) reste.

Après avoir branché l'alimentation, le nom de la calculatrice s'affichera sur l'écran de la calculatrice pendant 1 seconde, ainsi qu'un code à 6 chiffres représentant la date de la version du micrologiciel de la calculatrice.
Par exemple. "ET-57 201101" signifie la date du firmware (build) 11/1/2020.

La calculatrice dans la variante **ET-57B** (processeur de calculatrice **ET-58** reprogrammé) peut être éteinte en appuyant sur **2nd CLR** (fonction **OFF**) et allumée en appuyant sur **CLR**.
En appuyant sur **INV 2nd CLR** (fonction LCD) suivi d'un chiffre de **0** à **9**, le contraste de l'affichage peut être contrôlé.

5. Écarts par rapport à la **TI-57**

Bien que le logiciel de la calculatrice **ET-57** s'efforce d'obtenir une compatibilité maximale avec la calculatrice **TI-57** d'origine, des écarts peuvent se produire et certains programmes peuvent devoir être modifiés lors de l'importation.
Voici les écarts connus qui pourrait devoir être pris en compte :

Plus grande précision

La calculatrice **TI-57** originale fonctionne avec une précision interne de 11 chiffres (11 chiffres DCB de la mantisse, 2 chiffres de l'exposant, 1 chiffre du signe, 7 octets au total) et affiche un maximum de 8 chiffres de la mantisse.
La calculatrice **ET-57** calcule en interne avec une précision de mantisse de 17 chiffres (10 octets par nombre).
Elle stocke le résultat du calcul dans des registres de base (c'est-à-dire les registres **R0**..**R9** et **X**) avec une précision de 15 chiffres (9 octets). Lorsqu'elles sont stockées dans un registre étendu (**R10**..**R79**), les données sont arrondies à 13 chiffres (8 octets).

La calculatrice **TI-57** d'origine utilise la méthode CORDIC pour les calculs de fonctions, qui permet des calculs relativement rapides et faciles en utilisant uniquement des opérations de base (décalage, addition, soustraction) et des valeurs de table.
La méthode CORDIC est utilisée dans les calculatrices et en interne dans les processeurs.
En revanche, la calculatrice **ET-57** utilise la série Taylor pour les calculs, ce qui est plus adapté au type de processeur utilisé.

En conséquence de ce qui précède, la fonction **TI-57** d'origine calcule avec une précision de 9 à 10 chiffres, les 1 à 2 chiffres supplémentaires de la mantisse incluent des imprécisions de calcul. **ET-57** calcule des fonctions avec une précision de 15 chiffres (en interne, il calcule à 17 chiffres, en stockant dans le registre le résultat est arrondi à 15 chiffres valides).

Une plus grande précision n'est généralement pas un problème, elle peut apparaître, par exemple, avec un générateur de nombres aléatoires ou lors de la comparaison des résultats de calculatrices.

Répétition du calcul

La calculatrice **ET-57**, après avoir appuyé sur la touche [=] , répète la dernière opération arithmétique entrée, tandis que sur la **TI-57** d'origine, appuyer à plusieurs reprises sur la touche [=] provoque une indication d'erreur, que certains programmes utilisent pour indiquer une erreur. Dans de tels cas, il est nécessaire de fournir une indication d'erreur d'une autre manière, par exemple avec la séquence [CLR] [1/x] .

6. Format des nombres

Dans la calculatrice **TI-57** d'origine, un nombre est stocké dans des registres de 14 chiffres DCB D13 à D0. Un seul chiffre peut prendre des valeurs de 0 à 9. Les deux chiffres inférieurs, D1 et D0, contiennent l'exposant non signé, compris entre 00 et 99. Les 11 chiffres supérieurs, D12 à D2, contiennent les chiffres de la mantisse. La mantisse est toujours alignée à gauche afin que le chiffre D12 ne contienne pas de zéro. Le chiffre le plus élevé D13 contient les drapeaux de signe. Le bit 0 indique une mantisse négative, le bit 1 un exposant négatif et le bit 2 une mantisse inversée (signe de retenue du chiffre le plus élevé, mantisse sous forme négative). Cette méthode d'interprétation des nombres utilise le support du processeur pour les opérations DCB.

La calculatrice **ET-57** utilise également une interprétation DCB de la mantisse d'un nombre. Le format DCB assure un arrondi plus approprié des résultats pour une interprétation humaine. Par exemple. le nombre 0,1 est stocké dans le code DCB sous la forme du chiffre '1' avec un exposant de -1, sans perte de précision. En code binaire, un tel nombre s'exprimerait avec la mantisse 4CCCC... (nombre infini de chiffres), quand la simple écriture du nombre crée une petite erreur.

La mantisse n'est pas stockée dans la calculatrice **ET-57** sous forme absolue avec un signe séparé (comme dans la calculatrice d'origine), mais conserve la forme signée, avec une extension au chiffre le plus significatif. Un chiffre signé contient la valeur 0 (indiquant un nombre non négatif) ou 9 (indiquant un nombre négatif). La négation d'un nombre signifie le "complément décimal" des chiffres de la mantisse, ou le "complément à neuf" (= inversion) augmenté de 1.

L'exposant exprime à nouveau un exposant décimal, mais il est stocké dans le premier octet du nombre en tant que nombre binaire avec un biais de 128. Un exposant avec une valeur de 0 (ordre des uns) est symbolisé par la valeur binaire 128. Un l'exposant de 1 (dizaines) a une valeur de 129, un exposant de -1 (dixièmes) a la valeur de 127. L'exposant a une plage de valeurs binaires valides de 29 à 227, ce qui correspond à un exposant décimal de -99 à+99.
De plus, 3 cas particuliers de valeurs d'exposants sont utilisés : 0 indique zéro, 28 indique un dépassement vers les exposants négatifs (nombre trop petit) et 228 indique un dépassement vers les exposants positifs (nombre trop grand).

La calculatrice utilise 3 formats de nombres, différant par la précision de la mantisse :

1) Des nombres de 10 octets (précision de 17 chiffres) sont utilisés lors des calculs. Le premier octet est l'exposant sous forme binaire avec un biais de 128. Les 9 octets suivants contiennent la mantisse sous forme signée, des chiffres supérieurs aux chiffres inférieurs. Cela signifie 1 chiffre significatif et 17 chiffres significatifs. En affichant la mantisse sous forme HEX, le chiffre est affiché sous une forme lisible par l'homme, sous forme de nombres de gauche à droite.

La mantisse est normalisée de sorte que le premier chiffre (signe) contienne 0 (un nombre non négatif) ou 9 (un nombre négatif). Le deuxième chiffre (le chiffre le plus élevé de la mantisse) contient un nombre différent du chiffre signé - c'est-à-dire que la mantisse est alignée à gauche.

2) Les registres de base (**R0** à **R9**, ainsi que les registres **X** et LAST) occupent 9 octets (précision de 15 chiffres). Le premier octet est l'exposant, les 8 octets suivants contiennent 15 chiffres de mantisse et 1 chiffre de signe.

3) Les registres étendus (**R10** à **R79**) occupent 8 octets (précision 13 chiffres), avec une mantisse de 7 octets, soit 13 chiffres de mantisse et 1 chiffre de signe.

La mantisse du résultat peut être affichée à des fins de débogage avec les touches **INV +** (désactivée avec **INV -**). Les 16 chiffres du registre **X** sont affichés, sans l'exposant.

Exemples de nombres (au format HEX, incluant un exposant avec un biais de 128) :

3.14159265358979 -> 80 03 14 15 92 65 35 89 79
-3.14159265358979 -> 80 96 85 84 07 34 64 10 21
123.456 -> 82 01 23 45 60 00 00 00 00

Un test de trigonométrie célèbre peut être utilisé pour tester la précision de la calculatrice :

[9] [sin] [cos] [tan] [INV] [sin] [INV] [cos] [INV] [tan]

Si le calcul est correct, le résultat devrait être à nouveau le nombre 9.
Le calcul perd rapidement de sa précision et les écarts sont courants avec les calculatrices.
Pour la calculatrice **TI-57** d'origine, le résultat est 9,0047464 (précision à 3 chiffres), pour la calculatrice **ET-57**, le résultat du test est 8,9999999976 (précision à 9 chiffres).

Plus d'informations sur la précision des calculatrices :
http://www.datamath.org/Forensics.htm

7. Clavier

La calculatrice peut fonctionner soit en mode direct (exécution), lorsque les codes des touches sont exécutés immédiatement, soit en mode programmation, lorsque les codes des touches sont uniquement enregistrés dans le programme, mais pas exécutés.

La calculatrice est contrôlée par un ensemble de 40 touches disposées en 8 lignes et 5 colonnes.
Les lignes sont numérotées de haut en bas, dans l'ordre de 1 à 8.
Les colonnes sont numérotées de gauche à droite, avec des numéros de 1 à 5.
C'est avec cette numérotation, que les codes des touches sont stockés dans le programme.

Après avoir appuyé sur la touche [2nd], la fonction alternative de la touche suivante est utilisée, indiquée par le numéro de colonne 6 à 10 (le numéro 10 est remplacé par le numéro 0 dans le code).

Lors de l'écriture du programme dans la mémoire (à l'aide de la touche [LRN]), le code de la touche enfoncée est écrit dans le programme sous la forme d'une paire de chiffres, où le premier chiffre représente la rangée de la touche (généralement 1 à 9) et le second chiffre représente la colonne de la touche (typiquement 1 à 5 pour la fonction de base ou 6 à 0 pour une fonction alternative).

Les codes de touches numériques de [0] à [9] ne sont pas stockés dans le programme en utilisant la coordonnée de la touche, mais sous forme de valeur décimale de 00 à 09.

Remarque : Dans le texte du manuel, les noms des touches sont donnés sans aucun deuxième préfixe qui peut être nécessaire pour appeler la fonction de la touche. Par exemple. le code bouton [Rand] (numéro aléatoire) est appelé en appuyant sur les touches [2nd] et [3].

8. Indicateurs à l'écran

L'afficheur LCD contient 2 lignes de 16 caractères alphanumériques.
La première ligne sert à afficher les indicateurs, la deuxième ligne à afficher le nombre saisi et le résultat de l'opération.

Deg/**Rad**/**Grd** :
indication de l'unité d'angle en degrés, radians ou grades.
360° = 2*PI radians = 400 grades.
Le commutateur peut être changé avec les touches [Deg], [Rad] ou [Grad]

Fix 0 à **Fix 8** :
indique l'arrondi sélectionné des nombres de 0 à 8 décimales. Il est réglé avec les touches [Fix] [0] à [Fix] [8]

La séquence [INV] [Fix] ([Fix] [9] également la même signification) désactive l'arrondi du résultat affiché. Dans ce cas, l'arrondi n'est pas indiqué sur l'afficheur (il est remplacé par des espaces).

EE :
indique le mode exposant.
Après avoir appuyé sur [EE], le nombre s'affiche sous forme scientifique, sous forme de mantisse et d'exposant.
Le mode peut être annulé en appuyant sur [CLR] ou [INV] [EE]
Si le mode exposant est désactivé, rien n'est indiqué à l'écran.

2nd :
indique que vous avez appuyé sur la touche de fonction alternative [2nd] .
Si une touche est enfoncée après une pression sur [2nd], sa fonction alternative (affichée dans la deuxième rangée du clavier) sera exécutée à la place de sa fonction de base.
Si la touche [2nd] n'est pas enfoncée, ou si elle est enfoncée deux fois, la fonction alternative n'est pas active, la fonction de base du bouton est exécutée.
L'état de base n'est pas indiqué sur l'affichage (des espaces sont affichés à la position).

INV :
indique que vous avez appuyé sur la touche [INV] activant la fonction d'inversion.

Opération :
la dernière position de la 1ère ligne est destinée à indiquer l'opération arithmétique active : **+** addition, **-** soustraction, ***** multiplication, **:** division, \ modulo...

9. Éditeur de nombre

Le nombre entré, ainsi que les résultats du calcul, seront affichés sur la 2ème ligne de l'écran. La mantisse est affichée avec un maximum de 11 chiffres.

1 position est réservée au signe avant la mantisse. Un '**-**' apparaîtra ici pour les nombres négatifs, un espace sera laissé pour les nombres positifs.

L'exposant est affiché après la mantisse (si le mode exposant est actif). L'exposant est séparé de la mantisse par un signe **+** ou **-**. L'exposant est affiché sous forme de 2 chiffres.

Un point décimal fait partie de la mantisse. En mode exposant en notation scientifique (mantisse et exposant), le point décimal apparaît toujours après le premier chiffre. Si le mode exposant n'est pas actif, un point décimal s'affiche après le chiffre des unités.

La touche [CE] supprime le dernier caractère de la mantisse ou de l'exposant (selon l'endroit où les chiffres sont actuellement écrits).

La touche [EE] commence à saisir l'exposant.
Vous pouvez revenir à la saisie de la mantisse en appuyant sur la touche [.] (point) ou [INV] [EE]

La touche [EE] est également utilisé pour commencer à éditer le résultat affiché de l'opération. Cela peut être utilisé pour supprimer les chiffres cachés d'un nombre.

Exemple : arrondir un nombre sur 4 décimales

[π]		3.1415926536	exemple d'un nombre avec décimales
[Fix]	[4]	3.1416	l'affichage est arrondi à 4 décimales
[EE]		3.1416+00	démarre l'édition, coupe les chiffres masqués
[INV]	[EE]	3.1416	désactive le mode exposant
[INV]	[Fix]	3.1416	en désactivant l'arrondi, le résultat reste à 3.1416

10. Expressions numériques

Lors des calculs, le calculateur maintient la priorité des opérations en 3 étapes :

①. ^ puissance, √‾ racine carrée

②. * multiplication, : division, \ modulo

③. + addition, - soustraction

Les calculs sont d'abord évalués au niveau ① puissance et racine carrée, puis ② multiplication et division, et enfin ③ addition et soustraction.

N'importe quel nombre de parenthèses peut être utilisé dans une expression, jusqu'au niveau 7.

Après avoir effectué le calcul, vous pouvez répéter le calcul du niveau le plus bas en appuyant à nouveau sur la touche [=].
Entrez un nombre et appuyez sur = pour répéter l'opération. Le nombre saisi est utilisé comme premier opérande de l'opération, le deuxième opérande reste d'origine.

Remarque : Les calculs sont effectués en interne avec une précision de 17 chiffres. En stockant le résultat dans le registre **X**, le résultat est arrondi à 15 chiffres.

Exemple:

- 3 + 2 = 5
- 4 =6
- 1 0 =12
- 1 0 + 2 * 3 y^x 4 = 172 [*identique à 10 + (2*(3^4)) = 172*]

11. Adressage des Registres

La calculatrice contient 10 registres de base (marqués **R0** à **R9**) et 70 registres additionnels (marqués **R10** à **R79**), pour un total de 80 registres de données (**R0** à **R79**).
Les registres de base **R0** à **R9** ont une précision de mantisse de 15 chiffres. Les registres de base sont utilisés comme registres de travail principaux. Ils sont adressés par adressage direct, à l'aide des instructions
[STO] [RCL] [SUM] [Exc] [Prd] [Inc] suivies des index de registre [0] à [9].
Les registres de base sont également adressables par les opérations inverses
[INV] [SUM] , [INV] [Prd] , [INV] [Inc]

Les instructions [INV] [STO] , [INV] [RCL] , [INV] [Exc] ont la même fonction que les instructions sans [INV] , mais au lieu des registres de base **R0** à **R9**, elles adressent les registres additionnels **R10** à **R19**.

La plupart des registres de base ont une fonction supplémentaire :
R0 ... nombre d'éléments N (Statistiques), compteur de boucle **Dsz**
R1 ... somme de y (Statistiques)
R2 ... somme de y^2 (Statistiques)
R3 ... somme de x (Statistiques)
R4 ... somme de x^2 (Statistiques)
R5 ... somme de x*y (Statistiques)
R6
R7 ... registre **T**
R8 ... registre d'index pour l'adressage indirect
R9 ... registre d'index alternatif pour l'adressage indirect

Les registres de données additionnels **R10** à **R79** ont une précision réduite à 13 chiffres. En les utilisant, les données stockées sont raccourcies de 2 chiffres.

Les registres additionnels (**R10** à **R79**) ne peuvent pas être adressés par adressage direct, il est nécessaire d'utiliser l'adressage indirect :

STO* , RCL* , SUM* , Exc* , Prd* et Inc*

En adressage indirect, le numéro de registre ne fait pas partie de l'instruction, mais est lu dans le registre **R8** (registre d'adressage indirect). Les inverses sont traités de la même manière

Opérations d'adressage indirect précédées de INV :

Les instructions

INV STO* , INV RCL* , INV Exc*

ont la même fonction que les mêmes instructions sans **INV**, mais, dans ce cas, le registre d'index alternatif **R9** est utilisé à la place du registre d'index **R8**.

Remarque : Les registres de base **R0** à **R9** peuvent également être adressés par adressage indirect.

Exemple, inverser l'ordre du contenu du registre :

1) Remplissage des registres R10 à R79 avec les numéros 0 à 69

	Pgm 1	passage à l'espace programme 1 (et raz pointeur)
	LRN	activation du mode de programmation
00	Lbl 1	étiquette sous-programme 1 (remplissage des registres)
01	7 0 STO 0	préparation du compteur de registres (70 registres)
04	9 STO 8	préparation du registre d'index - 1 (pointeur vers R10-1)
06	CLR	mise à zéro registre X
07	Lbl 9	étiquette de début de boucle
08	Inc 8	incrémentation R8 (registre d'index)
09	STO*	stocke la valeur de X dans le registre indexé par R8
10	+ 1 =	incrémentation du registre X
13	Dsz	décrémente R0 et saute l'instruction suivante si R0=0
14	GTO 9	continuer la boucle si R0 n'est pas encore égal à 0
15	INV SBR	fin de programme

2) Inversion du contenu des registres (R10<->R79, R11<->R78,...)

16	Lbl 2	étiquette sous-programme 2 (inversion des registres)
17	3 5 STO 0	préparation du compteur de registres (70 registres / 2)
20	9 STO 8	préparation du premier registre d'index - 1
22	8 0 STO 9	préparation du deuxième registre d'index + 1
25	Lbl 8	étiquette de début de boucle
26	Inc 8	incrémentation R8 (premier registre d'index)
27	INV Inc 9	décrémentation R9 (deuxième registre d'index)
28	RCL*	lire la valeur du registre indexé par le registre R8
29	INV Exc*	échange avec le contenu du registre indexé R9
30	STO*	stocke la valeur dans le registre indexé par R8
31	Dsz	décrémente R0 et saute l'instruction suivante si R0=0
32	GTO 8	continuer la boucle si R0 n'est pas encore égal à 0
33	INV SBR	fin de programme

3) Affichage du contenu des registres **R10** à **R79**

34	Lbl 3	étiquette sous-programme 3 (affichage des registres)
35	7 0 STO 0	préparation du compteur de registres (70 registres)
38	9 STO 8	préparation du registre d'index - 1 (pointeur vers **R10**-1)
40	Lbl 7	étiquette de début de boucle
41	Inc 8	incrémentation **R8** (premier registre d'index)
42	RCL*	lire la valeur du registre indexé par le registre **R8**
43	Pause	pause pour afficher la valeur
44	Dsz	décrémente **R0** et saute l'instruction suivante si **R0**=0
45	GTO 8	continuer la boucle si **R0** n'est pas encore égal à 0
46	INV SBR	fin de programme

4) Test du programme

LRN	quitter le mode de programmation
SBR 1	remplir les registres **R10** à **R79** avec le numéro 0 à 69
INV RCL 9	afficher le contenu du registre **R19** (9)
SBR 3	vérifier le contenu des registres : les nombres 0 à 69 s'affichent
SBR 2	inversion des contenus des registres
INV RCL 9	afficher le contenu du registre **R19** (60)
SBR 3	vérifier le contenu des registres : les nombres 69 à 0 s'affichent

12. La programmation

L' écriture d'une séquence de touches dans la mémoire programme s'appelle un programme. Le programme transforme la calculatrice en un outil puissant.
La mémoire de programme se compose de 10 zones de programme indépendantes, commutées par des instructions Pgm avec les paramètres 0 à 9
Chaque zone de programme contient 50 pas de programme, donc un total de 500 pas de programme sont disponibles.
10 étiquettes Lbl , numérotées de 0 à 9 , peuvent être utilisées dans chaque espace de programme.
Des sous-programmes peuvent être appelés ou des sauts effectués entre les espaces de programme en utilisant l'instruction Pgm dans le programme, cette l'instruction ne change pas d'espace de programme de façon permanente, mais seulement pour une commande GTO ou SBR

Les programmes sont stockés dans la mémoire EEPROM du processeur dont le contenu est conservé même après la déconnexion de l'alimentation de la calculatrice.

Le mode de programmation est lancé avec la touche LRN.
Le contenu du programme est affiché sur deux lignes de l'afficheur. La ligne du bas à partir de la gauche indique l'espace programme courant (**Pgm0** à **Pgm9**), suivi du pointeur courant dans le programme, c'est-à-dire l'adresse **00** à **49**.
L'adresse est suivie du code numérique de l'instruction.
Ce code d'instruction se compose de 2 chiffres. Le premier chiffre représente la ligne avec les touches 1 à 8, le deuxième chiffre est la colonne avec les touches 1 à 5 ou, pour la fonction alternative, la colonne 6 à 0. Les touches numériques sont affichées avec le code **00** à **09**.
Le code d'instruction peut être suivi d'un paramètre de 0 à 9. Le code de l'instruction peut aussi être précédé du signe moins '**-**' signifiant la fonction inverse INV .
La ligne supérieure affiche le format "texte" de l'instruction.

Touches utiles à la programmation :

SST	(*Single Step*)	Incrémente le pointeur de programme de 1 ("pas suivant"). La touche **SST** peut également être utilisé en mode normal (exécution). Dans ce mode, après avoir appuyé sur **SST**, le code de l'instruction courante est exécuté et le pointeur de programme se positionne sur le pas suivant.
BST	(*Back Step*)	Décrémente le pointeur de programme de 1 ("pas précédent").
Ins	(*Insert*)	Insère une instruction **Nop** vide à la position actuelle du programme et décale le reste du programme.
Del	(*Delete*)	Supprime l'instruction à la position courante du programme et "remonte" la partie suivante du programme.
LRN	(*Learn*)	Quitte le mode d'édition de programme et ramène la calculatrice en mode d'exécution.
GTO	(*Go To*)	L'instruction **GTO** est normalement utilisé pour déplacer le pointeur de programme vers l'étiquette spécifiée de 0 à 9. Dans le mode "programmation", l'instruction ne peut pas être utilisée pour déplacer le pointeur, car le code correspondant serait stocké dans le programme. Il faut donc quitter le mode programmation en appuyant sur LRN, effectuer un saut GTO vers le label spécifié 0 à 9 et revenir en mode programmation en appuyant sur LRN. En plus de sauter à une étiquette, **GTO** vous permet de sauter à une adresse absolue dans le programme. Le saut s'effectue en appuyant sur INV GTO, suivi des 2 chiffres de l'adresse 00 à 49. L'instruction de saut à l'adresse absolue ne peut pas être mémorisée dans le programme, elle sert uniquement à déplacer le pointeur lors de la programmation et peut donc être saisie à la fois en exécution et en programmation. ***Remarque** : Le saut vers une adresse absolue se fait d'une manière différente (avec une séquence de touches différente) sur l' **ET-57** et la **TI-57** d'origine. Sur la **TI-57** d'origine, la touche **GTO** a été enfoncée en premier, puis la touche **INV** et enfin les 2 chiffres de l'adresse. Pour l' **ET-57**, il faut d'abord appuyer sur **INV**, puis **GTO** et enfin les 2 chiffres de l'adresse.*
RST	(*Reset*)	Semblable à **GTO**, ne peut pas être utilisé directement en mode de programmation, mais peut être utilisé en mode d'exécution pour ramener le pointeur de programme à l'adresse 00 dans l'espace de programme actuel.
R/S	(**Run**/*Stop*)	Démarre le programme ou arrête le programme (utilisé en mode exécution).

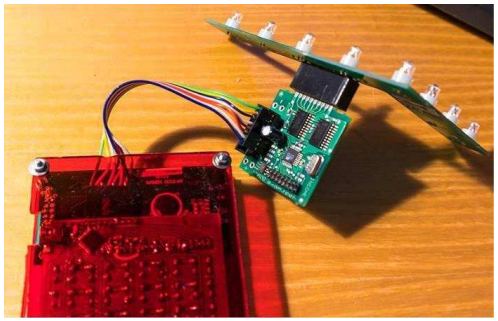
13. Périphériques et ports externes

La calculatrice **ET-57** permet la connexion d'un périphérique externe à l'aide du connecteur ISP, qui est autrement utilisé pour programmer le processeur de la calculatrice. La communication s'effectue via le protocole SPI. Le connecteur ISP est un connecteur KONPC-SPK-8 à 8 broches avec l'affectation des broches suivante :

- 1 **SCK** (horloge série, transmise par calculateur)
- 2 **MISO** (entrée de données dans la calculatrice, sortie de données de l'appareil)
- 3 **MOSI** (sortie de données de la calculatrice, entrée de données dans l'appareil)
- 4 clé d'orientation, broche manquante, aveuglée dans le connecteur de sorte qu'elle ne peut pas être insérée
- 5 **/RESET** (réinitialisation du processeur pendant la programmation ISP)
- 6 **GND** (masse, 0V)
- 7 **VCC** (alimentation, +5V)
- 8 inutilisé, connecté au +5V dans la calculatrice, mais à l'avenir il ne sera peut-être pas utilisé ou pourra être utilisé pour le signal SS.

Le câble de connexion connecte les mêmes broches à la fois dans la calculatrice et dans l'appareil (pas de croisement de signaux).
Remarque : Le signal SS du côté esclave doit être connecté à GND.

L'appareil peut être contrôlé à l'aide d'un réseau de ports adressables de 256 octets. Un nombre négatif -1 à -256 représentant l'adresse du port 0 à 255 est stocké dans le registre **R8** du calculateur L'instruction **STO*** envoie le nombre 0 à 255 au port sélectionné.
Inversement, l'instruction **RCL*** lit la valeur du port sélectionné sous la forme d'un nombre compris entre 0 et 255.



De même, les instructions **INV STO*** et **INV RCL*** peuvent être utilisées, dans lesquelles le registre **R9** avec l'adresse du port est utilisé à la place du registre **R8**. Les instructions autres que STO* et RCL* n'autorisent pas l'accès aux ports de l'appareil.

Si l'appareil n'est pas connecté ou dans un autre cas d'erreur de communication, lors de l'utilisation des instructions **STO*** et **RCL***, le programme s'arrêtera et indiquera une erreur ('***E***' clignotant). Pour les instructions **INV STO*** et **INV RCL***, l'erreur de communication est ignorée, le programme n'indique pas l'erreur et continue à fonctionner. En règle générale, lors de la mise au point du programme, les instructions **STO*** et **RCL*** sont utilisées en premier pour s'assurer que les problèmes de communication sont signalés. Après le débogage du programme, des instructions avec **INV** sont utilisées pour garantir que le programme fonctionnera sans interruption pendant une longue période, même en cas de pannes d'équipement à court terme.

La communication s'effectue à l'aide du protocole de communication SPI, avec une fréquence d'horloge de 250 kHz. Le calculateur agit comme un maître (unité de contrôle), l'appareil externe est un esclave (unité subordonnée). Le maître (calculateur) envoie des données série sur la ligne MOSI, du bit supérieur au bit inférieur. Il envoie un signal d'horloge à la ligne SCK, avec échantillonnage des données sur le front montant.

En même temps, l'esclave (appareil) renvoie des données sur la ligne MISO. Le transfert d'un octet (8 bits) prend 32 us. Le maître ajoute un délai de 10 us après l'envoi de chaque octet afin que l'esclave ait le temps d'évaluer l'octet reçu.

La synchronisation de la communication entre maître et esclave n'est pas contrôlée par le signal SS, mais par logiciel, en réinitialisant la réception côté esclave. Au début de chaque transmission, le maître envoie l'octet de synchronisation 0x53 (la lettre 'S', au niveau du bit 01010011b). Si le transfert n'est pas synchronisé, les données côté esclave sont décalées et l'esclave reçoit une valeur autre que 0x53. Dans un tel cas, l'esclave envoie l'octet 0x55 comme indication d'une erreur de synchronisation, réinitialise la connexion et après un délai de 50 us initialise une nouvelle connexion. Si l'esclave reçoit l'octet de synchronisation correct 0x53, il répond avec la même valeur de 0x53 et continue la comunication.

Le maître envoie plusieurs fois les octets de synchronisation 0x53. S'il reçoit une réponse 0x53 correcte de l'esclave, il continue la communication. S'il reçoit un octet de 0xff ou 0x00, il le traite comme un indicateur de périphérique non connecté et interrompt la communication avec une indication d'erreur. En cas d'autres réponses, il ajoute un délai de 100 us et retente la tentative de synchronisation.

Après une synchronisation réussie, le maître continue en envoyant la commande 0x52 (lettre 'R') pour lire le port de l'appareil ou 0x57 (lettre 'W') pour écrire le port sur l'appareil. Après la commande, le maître envoie un octet avec l'adresse de port de 0 à 255. Après avoir envoyé l'adresse, le maître envoie le troisième octet de la commande - données. Dans le cas d'une écriture sur le port, il enverra les données 0 à 255 pour écrire sur le port. Dans le cas d'une lecture depuis le port, le maître envoie un octet 0xFF, tandis que l'esclave envoie un octet, il répond avec un octet des données lues depuis le port.

Dans tous les autres cas, lorsque l'octet de l'esclave n'est pas spécifié, l'esclave répond en répétant l'octet reçu du maître (écho).

Exemple de communication (maître/esclave) :

- Le maître écrit l'octet 0x24 sur le port 0x01

0x53 / 0x ?? ... le maître envoie 5x commandes SYNC pour la synchronisation, la réponse de l'esclave est initialement indéfinie, elle est ignorée
0x53 / 0x53 ... l'esclave répond correctement avec 0x53, mais le maître ignore la réponse pour l'instant, car cela peut être le reste de la communication précédente
0x53 / 0x53
0x53 / 0x53
0x53 / 0x53 ... le maître détecte la réponse correcte 0x53, continue
0x57 / 0x53 ... le maître envoie une commande pour écrire 0x57, l'esclave continue d'écho
0x01 / 0x57 ... le maître envoie l'adresse de port 0x01, l'esclave a détecté une commande d'écriture, envoie l'écho de l'octet précédent et lit l'adresse
0x24 / 0x01 ... le maître envoie les données 0x24 à écrire sur le port

- Le maître relit les données du port 0x01

0x53 / 0x01 ... le maître envoie à nouveau 5x commandes SYNC pour la synchronisation
0x53 0x53 0x53 0x53 / 0x53 0x53 0x53 0x53 ... esclave connecté
0x52 / 0x53 ... le maître envoie une commande pour lire 0x52
0x01 / 0x52 ... le maître envoie l'adresse du port 0x01, l'esclave prépare les données du port pour l'envoi
0xFF / 0x24 ... le maître envoie 0xFF et lit les données esclaves 0x24

Exemple de code source pour gérer la communication Esclave :

```
u8 SPIData[256];        // SPI slave data
u8 SPIInx;      // index of SPI command
u8 SPICmd;      // 1st byte - SPI command
u8 SPIAddr;     // 2nd byte - SPI address

#define SPICMD_SYNC         0x53    // synchro command ('S')
#define SPICMD_READ         0x52    // read command ('R')
#define SPICMD_WRITE        0x57    // write command ('W')
#define SPICMD_ERR          0x55    // invalid synchronization ('U')

ISR(SPI_STC_vect)     // SPI interrupt
{
        u8 d = SPDR;   // read data byte
        if (SPIInx == 0) // receive command
        {
                // check known commands
                if (    (d != SPICMD_SYNC) &&
                        (d != SPICMD_READ) &&
                        (d != SPICMD_WRITE))
                {
                        // error, try to re-synchronize
                        SPDR = SPICMD_ERR; // report error to master
                        SPI_Term(); // terminate SPI
                        _delay_us(50); // short delay 50 us
                        SPI_SlaveInit(); // re-initialize SPI
                        return;
                }

                // shift to next byte of the command
                if (d != SPICMD_SYNC)
                {
                        SPICmd = d;
                        SPIInx = 1; // next index = address
                }
        }
        else if (SPIInx == 1)     // receive address
        {
                SPIAddr = d; // save address
                SPIInx = 2; // next index = data

                // read data
                if (SPICmd == SPICMD_READ) SPDR = SPIData[d];
        }
        else    // receive data
        {
                // next index = command
                SPIInx = 0;

                // write data
                if (SPICmd == SPICMD_WRITE) SPIData[SPIAddr] = d;
        }
}
```

14. Touches et instructions

Chaque intruction a un code DCB de programme, un intitulé et une séquence d'appuis sur une ou plusieurs touches pour l'exprimer.

00...09 [0] [9] 0...9 - Chiffres de base

Les chiffres de base sont utilisés pour entrer des chiffres dans la plage de 0 à 9. Ils sont utilisés pour entrer la mantisse d'un nombre, entrer l'exposant, le numéro de registre de mémoire, le numéro d'étiquette et autres.

Les numéros sont stockés dans le programme avec le code 00 à 09.

10 [2nd] [CLR] OFF - Éteindre la calculatrice (ET-57B)

La séquence **2nd CLR** (**OFF**) sert à éteindre la calculatrice.
La calculatrice peut être rallumée en appuyant sur la touche **CLR** seule.

En spécifiant le préfixe **INV** avant l'instruction **OFF**, le contraste de l'affichage LCD peut être réglé.
L'instruction demande comme paramètre un code numérique de 0 à 9.
0 définit le contraste d'affichage le plus faible (police claire sur fond clair), **9** définit le contraste d'affichage le plus élevé (police sombre sur fond sombre).

Les fonctions **OFF** et **INV OFF** ne sont disponibles que pour la variante **ET-57B** de la calculatrice.
La calculatrice **ET-57** ne peut pas être éteinte avec un bouton, et le contraste de l'affichage ne peut pas non plus être contrôlé.

11 [2nd] 2nd - Fonction alternative

La touche **2nd** est utilisé pour changer la signification de la touche suivante en une fonction alternative.
Après avoir appuyé sur **2nd**, la fonction alternative du bouton suivant est alors exécutée.
Un deuxième appui sur **2nd** revient aux fonctions de base.
(annule le premier appui sur **2nd**).

Le code de la touche **2nd** (**11**) n'est pas enregistré dans le programme, c'est le code alternatif de la touche suivante qui est alors enregistré.

Exemple:
2 lnx ... calcule le logarithme naturel du nombre 2 [0.6931...]
2 2nd lnx ... logarithme décimal du nombre 2 (instruction **log**) [0.3010...]

12 [INV] INV - Inverse de fonction

La touche **INV**, pressée avant une autre touche, fera que cette autre touche aura sa fonction inversée.
Dans certains cas particuliers, **INV** n'entrainera pas l'inverse de la fonction suivante mais une fonction alternative supplémentaire.
Un deuxième appui sur **INV** ramène aux fonctions de base.
(annule le premier appui sur **INV**).
Le code du bouton **INV** n'est pas enregistré dans le programme. Il est stocké sous la forme d'un indicateur d'instruction par le signe moins (**-**) avant le code d'instruction.
Certaines instructions n'acceptent pas le préfixe **INV**, celui-ci est alors ignoré et le code de l'instruction est stocké sans le préfixe dans le programme.

Exemple:
1 0 sin ... calcule le sinus de 10 [0,1736...]
0 . 1 2 INV sin ... calcule l'arcsinus de 0.12 [0.3464...]

13 [lnx] lnx - Logarithme népérien et exposant

lnx calcule le logarithme népérien du nombre affiché. Ce logarithme naturel utilise la constante d'Euler comme base avec la valeur 2,718281828459. Si le bouton **INV** est enfoncé en premier, la fonction inverse, l'exposant naturel, est exécutée.
L'argument de la fonction **lnx** doit être un nombre positif non nul. Dans le cas de zéro, l'affichage clignotera avec la valeur -9.9999+99, comme indication d'erreur.
Pour un nombre négatif, la valeur absolue du nombre est calculée et l'affichage clignote à nouveau avec une indication d'erreur.

L'argument de la fonction **INV lnx** peut être un nombre positif ou négatif, compris entre -227 et +227 environ. Un nombre en dehors de cette plage entraînera un débordement des données et indiquera une erreur.

Exemple:
5 lnx ... calcule le logarithme népérien de 5 [1.6094...]
5 INV lnx ... calcule l'exposant naturel de 5 [148.413...]

14 [CE] CE - Correction d'erreur de saisie

CE peut être utilisé pour annuler l'indication d'erreur '**E**', manifestée par le clignotement de l'affichage.
Lors de la saisie d'un nombre, le dernier caractère saisie est supprimé en appuyant sur la touche **CE**.
En notation scientifique, si la mantisse est en cours de saisie, le dernier caractère de la mantisse est supprimé. Si l'exposant est en cours de saisie, le dernier caractère de l'exposant est supprimé. Si un exposant avec une valeur de 0 est supprimé, l'exposant est annulé et la saisie revient à la mantisse.

15 [CLR] CLR - Effacement de l'affichage

CLR effectue plusieurs opérations d'initialisation.
Il réinitialise les opérations arithmétiques démarrées, réinitialise l'indication d'erreur, désactive le mode exposant **EE**, réinitialise le registre **X** et commence à éditer un nouveau nombre avec une valeur par défaut à 0.
La touche **CLR** ne réinitialise pas le registre **T** ni les registres de données.
Dans la variante de calculatrice **ET-57B**, la touche **CLR** est utilisée pour allumer (ou éteindre) la calculatrice.

18 [2nd] [lnx] log - Logarithme décimal et exposant

log permet de calculer le logarithme décimal du nombre affiché.
Le logarithme décimal utilise comme base le nombre 10. Si la touche **INV** est enfoncée en premier, la fonction inverse, l'exposant décimal, est exécutée.
L'argument de la fonction **log** doit être un nombre positif non nul. Dans le cas de zéro, l'affichage clignotera avec la valeur -9.9999+99, comme indication d'erreur.
Pour un nombre négatif, la valeur absolue du nombre est calculée et l'affichage clignote à nouveau avec une indication d'erreur.
L'argument de la fonction **INV log** peut être à la fois un nombre positif et un nombre négatif, compris entre -99 et +99. Un nombre en dehors de cette plage entraînera un débordement des données et une indication d'erreur.

Exemple:
5 log ... calcule le logarithme décimal de 5 [.69897...]
5 INV log ... calcule l'exposant décimal de 5 [100000]

19 [2nd] [CE] C.t - Effacer le registre T

La touche **C.t** peut être utilisée pour effacer le registre **T** (c'est-à-dire le registre **R7**).

La saisie du préfixe **INV** avant la touche **C.t** **efface tous les registres** **R0** à **R79** (y compris le registre **T**).

20 [2nd] [1/x] tan - Tangente

La fonction **tan** calcule la tangente d'un angle. L'angle est saisi dans les unités définies par les commutateurs **Deg**, **Rad** ou **Grad**.
La saisie du préfixe **INV** avant l'instruction **tan** exécutera la fonction opposée (arctangent).
Le résultat est un angle dans la mesure angulaire actuellement définie.

Exemple:
5 0 tan ... calcule la tangente de 50 [1.19175...]
5 0 INV tan ... calcule l'arctangente de 50 [88.85423...]

21 [LRN] LRN - Programmation

LRN active ou désactive le mode de programmation.

22 [x<>t] x<>t - Echange des registres X et T

Avec la touche **x<>t**, il est possible de commuter les registres **X** et **T**.
Le registre **X** est le registre de travail et également le contenu de l'affichage.
Le registre **T** est un registre auxiliaire (temporaire), correspond au registre **R07**. Il est utilisé pour comparer des nombres et pour convertir des coordonnées polaires et cartésiennes.
Le registre **X** est réinitialisé par la touche **CLR**.
Le registre **T** est réinitialisé par la touche **C.t.**

23 [x²] x^2 - Carré d'un nombre

La fonction **x^2** calcule le carré d'un nombre, ou le multiple d'un nombre par lui-même.

24 [√x] √x - Racine carrée d'un nombre

√x permet de calculer la racine carrée d'un nombre. Le nombre ne doit pas être négatif. Si un nombre négatif est calculé, la racine carrée de la valeur absolue du nombre est calculée et l'indication d'erreur '**E**' est activée (l'affichage clignote).

25 [1/x] 1/x - Inverse d'un nombre

La fonction **1/x** permet de calculer l'inverse d'un nombre.
Si le nombre est zéro, la valeur 9,9999+99 s'affiche et l'indication d'erreur '**E**' est activée (l'affichage clignote).
Cette application de **0 1/x** est souvent utilisé dans les programmes pour activer l'indication d'erreur et pour signaler un fonctionnement anormal du programme.

26 [2nd] [LRN] D.MS - Conversions minutes/secondes

L'instruction **D.MS** est utilisée pour convertir le temps exprimé en heures/minures/secondes ou l'angle exprimé degrés/minutes/secondes en un nombre décimal. Le nombre d'origine ***HH.MMSS*** ou ***DD.MMSS***, est saisi avec le nombre d'heures, ou de degrés, dans la position entière, puis avec le nombre de minutes dans les deux premières décimales et enfin avec le nombre de secondes dans les deux décimales suivantes.
Le résultat de la fonction est un nombre décimal représentant le nombre d'heures, ou de degrés, exprimé sous forme d'un nombre décimal ***DD.DDDD***.
En spécifiant le préfixe **INV** avant l'instruction **D.MS**, l'opération inverse est effectuée - le temps ou l'angle exprimé à l'aide d'un nombre décimal est converti en heures/minures/secondes ou degrés/minutes/secondes.
Le nombre décimal ***DD.DDDD*** représentant les heures ou les degrés sera restitué sous la forme ***HH.MMSS*** ou ***DD.MMSS***, avec le nombre d'heures, ou de degrés, dans la partie entière, et le nombre de minutes dans les deux premières décimales suivi du nombre de secondes dans les deux décimales suivantes. Si le résultat n'est pas un nombre entier de secondes, les décimales des secondes sont complétées par des chiffres décimaux supplémentaires.

Exemple :

1 2 . 3 0 2 3 D.MS ... conversion en décimales [12.50638...]
+ 3 . 4 5 1 2 D.MS ... conversion en décimales [3.75333...]
= INV D.MS ... temps résultant [16.1535]
Somme de temps :
12 heures, 30 minutes et 23 secondes
+ 3 heures, 45 minutes et 12 secondes
= 16 heures, 15 minutes et 35 secondes

27 [2nd] [x<>t] P->R - Conversion polaire/cartésien

P->R convertit les coordonnées de l'expression polaire (rayon/angle) en coordonnées cartésiennes (abscisse et ordonnée).
Avant l'opération, le registre **T** (c'est-à-dire le registre auxiliaire **R7**) contient le rayon et le registre **X** (contenu de l'affichage) contient l'angle. L'angle est donné dans la mesure angulaire sélectionnée (**Deg**, **Rad** ou **Grad**).
Après l'opération, le registre **T** (registre auxiliaire **R7**) contient l'abscisse X, le registre **X** (contenu d'affichage) contient l'ordonnée Y.
En spécifiant le préfixe **INV** avant l'instruction **P->R**, l'opération inverse est effectuée, la conversion des coordonnées cartésiennes en polaires.
Avant l'opération, le registre **T** contient l'abscisse X, le registre **X** (affichage) contient l'ordonnée Y.
Après l'opération, le registre **T** contient le rayon et le registre **X** (affichage) contient l'angle. L'angle est donné dans la mesure angulaire sélectionnée.

Exemple :
1 0 x<>t ... saisie du rayon 10 dans le registre **T**
3 0 ... saisie de l'angle de 30° dans le registre **X**
P->R ... conversion des coordonnées polaires en cartésiennes. Y = 5
x<>t ... échange **X** et **T** affiche les coordonnées X = 8.6602...

8.6602 ... x<>t saisie abscisse X
5 ... saisie ordonnée Y
INV P->R ... conversion cartésien en polaire, angle = 30°
x<>t ... échange X et T, affiche le rayon 10

28 [2nd] [x^2] sin - Sinus

La fonction **sin** calcule le sinus d'un angle. L'angle est saisi dans les unités définies par les commutateurs **Deg**, **Rad** ou **Grad**.
La saisie du préfixe **INV** avant l'instruction **sin** exécutera la fonction opposée arcsinus.
Le résultat est un angle dans la mesure angulaire définie.
L'angle calculé par la fonction arcsinus est compris entre -90° et +90°. La valeur d'entrée de la fonction arc sinus doit être comprise entre -1 et +1. S'il est en dehors de la plage spécifiée, l'affichage est limitée à la plage valide et une erreur "**E**" est indiquée (l'affichage clignote).

29 [2nd] [√x] cos - Cosinus

La fonction **cos** calcule le cosinus d'un angle. L'angle est saisi dans l' unité définie **Deg**, **Rad** ou **Grad**.
La saisie du préfixe **INV** avant l'instruction **cos** exécute la fonction opposée arccosinus. Le résultat est un angle dans la mesure angulaire définie.
L'angle calculé par la fonction arccosinus est compris entre 0° et +180°. La valeur d'entrée de la fonction arccosinus doit être comprise entre -1 et +1. S'il est en dehors de la plage spécifiée, l'affichage est limitée à la plage valide et une erreur "**E**" est indiquée (l'affichage clignote).

30 [2nd] [y^x] PI - Nombre de Ludolf

La touche **pi** est utilisée pour entrer la constante "nombre de Ludolf", qui a une valeur de 3,14159265358979.

31 [SST] SST - Avance d'un pas dans le programme

La touche **SST** (*Single Step*) augmente le pointeur d'adresse de programme de 1 en mode de programmation.
En mode exécution, l'instruction de programme, sur laquelle le pointeur est positionné, est exécutée, ce qui permet d'exécuter le programme pas à pas à des fins de débogage.
Attention : dans ce cas de test pas à pas du programme si un sous-programme est appelé, le retour du sous-programme ne peut pas se produire correctement (la calculatrice ne mémorise pas l'adresse de retour du sous-programme).

32 [STO] STO - Enregistrement dans un registre

STO (*Store*) permet de stocker le nombre affiché dans le registre de données **R0** à **R9**. Un numéro de registre de 0 à 9 est entré comme paramètre d'instruction.
Le préfixe **INV** avant l'instruction **STO** remplit une fonction similaire, mais au lieu des registres **R0** à **R9**, le nombre est stocké dans les registres **R10** à **R19**. Les registres **R10** à **R19** appartiennent au groupe des registres étendus avec une précision réduite à 13 chiffres. L'enregistrement supprime les 2 derniers chiffres de la mantisse.

33 [RCL] RCL - Rappel du contenu d'un registre

RCL (*Recall*) est utilisé pour rappeler un nombre du registre de données **R0** à **R9** vers l'affichage. Un numéro de registre de 0 à 9 est entré comme paramètre d'instruction.
Placer le préfixe **INV** avant l'instruction **RCL** exécute une fonction similaire, mais au lieu du registre **R0** à **R9**, le nombre est lu à partir du registre **R10** à **R19**. Les registres **R10** à **R19** appartiennent au groupe des registres étendus avec une précision réduite à 13 chiffres. La mantisse est complétée par 2 chiffres 0 à la fin.

34 [SUM] SUM - Addition dans un registre

SUM permet d'ajouter le nombre affiché (registre **X**) au registre **R0** à **R9**. Un numéro de registre de 0 à 9 est entré comme paramètre d'instruction.
En spécifiant le préfixe **INV** avant l'instruction **SUM**, la fonction opposée est effectuée - en soustrayant un nombre du registre de données **R0** à **R9**.

35 [y^x] y^x - Puissance et racine

L'instruction **y^x** élève le nombre Y (le premier opérande, dans la pile d'opérations) à la puissance exprimée par un autre nombre X (le deuxième opérande, le nombre sur l'affichage).
Si le préfixe **INV** est pressé en premier, l'opération inverse, la racine Xième, est effectuée. Le premier opérande de Y doit être un nombre non négatif. S'il s'agit du niveau le plus bas de l'expression, le calcul peut être répété pour un autre premier opérande en appuyant plusieurs fois sur **=**.

Exemple:
3 y^x 7 ... élévation de 3 à la puissance 7 [2187]
2187 INV y^x 7 ... racine 7ème de 2187 (2187^(1/7)) [3]

36 [2nd] [SST] Pause - Délai d'attente

La commande **Pause** arrête l'exécution du programme pendant 0,25 secondes et affiche le contenu du registre **X**.

37 [2nd] [STO] Ins - Insertion d'un pas dans le programme

La touche **Ins**, utilisée en mode programmation, insère une instruction **Nop** vide à la position du pointeur du programme. Les pas suivants sont décalés.

38 [2nd] [RCL] Exc - Échange X avec un registre

Exc permet d'échanger le nombre affiché (registre **X**) avec le contenu du registre de données **R0** à **R9**. Un numéro de registre de 0 à 9 est entré comme paramètre d'instruction.
Placer le préfixe **INV** avant l'instruction **Exc** exécute une fonction similaire, mais au lieu d'utiliser les registres **R0** à **R9**, le nombre est permuté avec les registres **R10** à **R19**. Les registres **R10** à **R19** appartiennent au groupe des registres étendus avec une précision réduite à 13 chiffres. Lors de la sauvegarde, les 2 derniers chiffres sont supprimés de la mantisse, et lors du chargement, la mantisse est complétée par 2 chiffres 0 à la fin.

39 [2nd] [SUM] Prd - Multiplier/diviser dans un registre

Prd permet de multiplier les registres de données **R0** à **R9** par le nombre affiché (registre **X**). Un numéro de registre de 0 à 9 est entré comme paramètre d'instruction.
En entrant le préfixe **INV** avant l'instruction **Prd**, la fonction inverse est effectuée en divisant le registre de données **R0** à **R9** par le nombre affiché.

40 [2nd] [÷] |x| - Valeur absolue

La fonction **|x|** ajuste le nombre à la valeur absolue (supprime le signe négatif du nombre).
Si le préfixe **INV** est spécifié avant d'appuyer sur **|x|**, la valeur absolue n'est pas appliquée mais un test de signe est effectuée sur le nombre. Si ce nombre est inférieur à 0, le résultat de l'opération est -1, si ce nombre est supérieur à 0, le résultat est +1. Si le nombre est 0, il reste 0.

41 [BST] BST - Recul d'un pas dans le programme

La touche **BST** (*Back Step*) en mode programmation décrémente le pointeur d'adresse de programme de 1.

42 [EE] EE - Mode Exposant

Appuyer sur **EE** pour activer le mode exposant.
Si la touche est enfoncée lors de la saisie d'un nombre, il passe à la saisie de l'exposant. En même temps, le mode d'affichage en notation scientifique avec un exposant est activé.
Si la touche est enfoncé en dehors de la saisie d'un nombre, le mode d'affichage en notation scientifique avec un exposant est activé et la saisie de l'exposant du nombre est lancée. Cette fonction est souvent utilisée pour supprimer les chiffres masqués d'un nombre, car lorsque vous commencez à saisir, seuls les chiffres affichés sont écrits dans l'afficheur, et non la valeur exacte complète du nombre.
Appuyez sur le préfixe **INV** avant d'appuyer sur **EE** pour quitter le mode d'affichage des exposants. Une autre façon de quitter le mode d'affichage des exposants consiste à appuyer sur la touche **CLR**.

Exemple:
pi Fix 4 EE INV EE INV Fix ... arrondit le nombre PI à 4 décimales [3.1416]

43 [(] (- Parenthèse gauche

Le caractère **(** ouvre le calcul d'une partie de l'expression.
Les parenthèses peuvent être utilisées jusqu'à 7 niveaux.

44 [)]) - Parenthèse droite

Le caractère **)** ferme le calcul d'une partie de l'expression.

45 [÷] : - Division

Le signe **:** permet de diviser le premier opérande par le deuxième opérande. S'il s'agit du niveau le plus bas de l'expression, le calcul peut être répété pour un autre premier opérande en appuyant plusieurs fois sur **=**.
Appuyer sur le préfixe **INV** avant d'appuyer sur **:** exécute la fonction inverse soit l'opération modulo mod qui restitue le reste après division. L'opération modulo divise le premier opérande Y (dans la pile) par le deuxième opérande X (sur l'affichage), convertit le résultat en un entier, multiplie le deuxième opérande X par celui-ci et soustrait du premier opérande Y. Le résultat est le reste après la division. Le résultat a le même signe que le premier opérande.

Exemple :
2 . 2 : 0. 5 = ... division de 2.2 par 0.5 [4.4]
2 . 2 INV : 0 . 5 = ... calcul du reste de 2,2 divisé par 0,5 [0,2]
2 . 2 +/- INV : 0 . 5 = ... calcul du reste de -2,2 divisé par 0,5 [-0,2]
2 . 2 INV : 0 . 5 +/- = ... calcul du reste de 2,2 divisé par -0,5 [0,2]
2 . 2 +/- INV : 0 . 5 +/- = ... calcul du reste de -2,2 divisé par -0,5 [-0,2]

46 [2nd] [BST] Nop - Aucune opération

La commande **Nop** (*No Operation*) est une commande vide qui n'effectue aucune opération. Elle sert uniquement à remplir un pas inutilisé dans le programme.

47 [2nd] [EE] Del - Suppression du pas courant

La touche **Del** (*Delete*), utilisée en mode programmation, supprime un pas de la position courante du programme. Les pas suivants sont alors remontés.

48 [2nd] [(] Fix - Nombre de décimales

À l'aide de la touche **Fix**, le nombre affiché à l'écran est arrondi au nombre de décimales spécifié. Le nombre 0 à 8 est saisi en paramètre, représentant le nombre de décimales après la virgule 0 à 8.
En mode arrondi, le nombre est complété à partir de la droite par des zéros, jusqu'au nombre spécifié de décimales. La saisie de la séquence **INV Fix** ou **Fix 9** désactive l'arrondi. Dans ce cas, le nombre est affiché en toute précision et les zéros non significatifs à la fin sont supprimés.
L'arrondi n'affecte que l'affichage du nombre. En interne, le nombre (registre **X**) continue a être mémoriser en entier. S'il est nécessaire de supprimer réellement les chiffres cachés, cela peut être fait en utilisant la touche **EE** (voir 42 Mode exposant, **EE**).
Le mode d'arrondi défini affecte également la manière dont les très petits nombres sont affichés. Si l'arrondi est activé et que le mode exposant n'est pas activé, l'écran affiche des zéros pour les petits nombres, même si les chiffres valides ont dépassé le bord droit de l'écran. Si l'arrondi n'est pas activé, la calculatrice passera à l'affichage de l'exposant si l'exposant est inférieur à -3.

49 [2nd] [)] Int - Partie entière

La touche **Int** est utilisée pour supprimer les chiffres après la virgule décimale du nombre ou pour réduire le nombre à un nombre entier.
La fonction a la même signification que l'arrondi vers zéro.
Si le préfixe **INV** est utilisé avant la commande **Int**, la fonction inverse est exécutée (**Frac**) en supprimant la partie entière du nombre et en ne conservant que la partie décimale.

Exemple :
2. 3 int ... partie entière de 2.3 [2]
2. 3 +/- Int ... partie entière de -2.3 [-2]
2. 3 INV Int ... partie décimale de 2.3 [0.3]
2. 3 +/- INV Int ... partie décimale de -2.3 [-0.3]

50 [2nd] [×] Deg - Degrés

La touche **Deg** commute les calculs des fonctions trigonométriques en degrés (un angle plein est de 360°).

51 [GTO] GTO - Saut vers label (ou adresse)

GTO permet d'effectuer un saut inconditionnel dans un programme. Il a comme paramètre un code numérique de 0 à 9 correspondant à un label (**Lbl**) du programme. En entrant la commande **Pgm** dans le programme avant la commande **GTO**, un saut à l'étiquette dans une autre zone de programme peut être effectué.
Lorsque l'instruction **GTO** est utilisée en mode exécution, le pointeur de programme est positionné sur l'étiquette sélectionnée.
En appuyant sur le préfixe **INV** avant l'instruction **GTO**, le pointeur dans le programme peut être déplacé vers une adresse absolue, qui est entrée sous la forme d'un code numérique à 2 chiffres de 00 à 49. Cette fonction ne peut pas faire partie du programme, elle est exécutée immédiatement , et permet de déplacer le pointeur du programme aussi bien en mode exécution qu'en mode programmation.
La touche **GTO** a une autre fonction spéciale : si vous la maintenez enfoncée pendant que le programme est en cours d'exécution, l'affichage fera défiler le contenu de l'affichage (registre **X**) sur la ligne du bas, et l'adresse du pas exécutée sur la ligne supérieure. Cependant, ce suivi ralentit considérablement le programme.

55 [×] x - Multiplication

La touche **x** permet de multiplier un premier opérande par un deuxième opérande. S'il s'agit du niveau le plus bas de l'expression, le calcul peut être répété pour un autre premier opérande en appuyant plusieurs fois sur **=**.

56 [2nd] [GTO] Dsz - Boucle de programme

L'instruction **Dsz** est utilisée pour exécuter une séquence de programme itérativement en utilisant une boucle selon un nombre de passages précisé dans le registre **R0**.
La fonction **Dsz** consiste dans le fait qu'elle décrémente (diminue de 1) le registre **R0** et arrête de boucler dès qu'il atteint zéro, elle ignore la commande suivante et continue l'opération au pas d'après. Si zéro n'est pas atteint (c'est-à-dire que la boucle n'est pas encore terminée), la commande suivant la commande **Dsz** est exécutée. En règle générale, **Dsz** est suivi de la commande **GTO**, qui renvoie au début de l'étiquette de la boucle.

Si le préfixe **INV** est donné avant l'instruction **Dsz**, le sens opposé de l'instruction est exécuté - l'instruction suivante est ignorée si le résultat de décrémentation n'est pas nul. Cette variante est généralement utilisée au début de la boucle. Lorsque le compteur de boucle atteint zéro, l'instruction suivante, qui est généralement une instruction de boucle **GTO**, est exécutée,
Dsz gére le registre **R0** et fonctionne plus précisément de la manière suivante :
Si la valeur contenue dans le registre **R0**

- est supérieure à 0 avant l'opération, la valeur du registre est diminuée de 1.
- est inférieure à 0, la valeur est augmentée de 1.
- est égale à 0, la valeur reste inchangé.

Si le résultat de l'opération est zéro ou si l'opération a franchi la limite zéro, l'opération pour zéro est effectuée selon la fonction sélectionnée.
Ce qui signifie que si le résultat de décrémentation/incrémentation n'a pas atteint zéro, la boucle se répète (l'instruction suivante est exécutée). Avec le préfixe **INV**, un saut est effectué au contraire si le résultat de l'opération atteint zéro (ou le dépasse).

Exemple *(factorielle d'un nombre)* ***:***

[RST] [LRN]		... activation du mode de programmation
00	[Lbl] [1]	... Label de début de programme
01	[STO] [0]	... stocke le nombre saisi dans le registre **R0**
02	[1]	... initialisation pour le calcul du produit
03	[Lbl] [9]	... label du début de la boucle
04	[(] [CE] [×] [RCL] [0] [)]	... multiplie le registre **X** par **R0**
09	[Dsz] [GTO] [9]	... décrémente **R0**, teste si zéro et boucle sinon
11	[INV] [SBR]	... fin de programme
[LRN]		... sortie du mode de programmation
[1] [2] [SBR] [1]		... test, factorielle de 12 [479001600]

57 [2nd] [7] STO* - Stockage indirect dans un registre

STO* stocke le contenu du registre **X** (affichage) dans un registre de données de la même manière que l'instruction **STO**.
Par contre, au lieu d'utiliser un paramètre d'instruction comme numéro de registre destinataire, le numéro de registre destinataire est contenu dans le registre **R8**.
Tous les registres de données **R0** à **R79** peuvent être adressés indirectement de cette manière.
Si le préfixe **INV** est donné avant l'instruction, le numéro de registre destinataire est contenu dans le registre **R9** (index alternatif).
Les registres **R10** à **R79** appartiennent au groupe des registres étendus avec une précision réduite à 13 chiffres.
L' enregistrement supprime les 2 derniers chiffres de la mantisse.
La saisie de l'adresse -1 à -256 (nombre négatif) dans le registre **R8** (ou **R9**) avec l'instruction **STO*** (ou **INV STO***) envoie un octet de valeur 0 à 255 (nombre de l'afficheur) au port 0 à 255 (selon le contenu du registre **R8**, **R9**) vers le dispositif externe connecté. (Voir Périphériques et ports externes).
Cette fonction ne peut pas être utilisée avec la variante de calculatrice **ET-57B**.

58 [2nd] [8] RCL* - Rappel indirect d'un registre

RCL* rappelle un nombre d'un registre de données de la même manière que l'instruction **RCL**.
Par contre, au lieu d'utiliser un paramètre d'instruction comme numéro du registre d'origine du nombre à récupérer, le numéro du registre d'origine est contenu dans le registre **R8**.
Tous les registres de **R0** à **R79** peuvent être adressés indirectement de cette manière.
Si le préfixe **INV** est donné avant l'instruction, le numéro de registre d'origine est contenu dans le registre **R9** (index alternatif).
Les registres **R10** à **R79** appartiennent au groupe des registres étendus avec une précision réduite à 13 chiffres.
La mantisse est complétée par 2 chiffres 0 à la fin.
La saisie de l'adresse -1 à -256 (nombre négatif) dans le registre **R8** (ou **R9**) avec l'instruction **RCL*** (ou **INV RCL***), lit un octet avec la valeur 0 à 255 du port 0 à 255 (selon le contenu du registre **R8**, **R9**) d'un appareil externe connecté. (Voir Périphériques et ports externes).
Cette fonction ne peut pas être utilisée avec la variante de calculatrice **ET-57B**.

59 [2nd] [9] SUM* - Addition indirecte dans un registre

L'instruction **SUM*** ajoute (ou soustrait avec **INV**) un nombre au registre destinataire de la même manière que l'instruction **SUM**, seulement au lieu du paramètre d'instruction, le numéro de registre destinataire est contenu dans le registre **R8**.
Les registres **R10** à **R79** appartiennent au groupe des registres étendus avec une précision réduite à 13 chiffres.
La mantisse est complétée par 2 chiffres 0 à la fin.

60 [2nd] [-] Rad - Radians

La touche **Rad** commute les calculs des fonctions trigonométriques en radians (un angle plein est de 2*pi rad).

61 [SBR] SBR - Appel sous-programme

La touche **SBR** (*Subroutine*) est utilisé pour appeler un sous-programme en utilisant en paramètre le code numérique de 0 à 9 du label appelé.
En spécifiant la commande **Pgm** dans le programme avant la commande **SBR**, un sous-programme d'un autre espace programme est appelé.
Si l'instruction **SBR** est utilisée en mode exécution, le sous-programme est exécuté immédiatement.
Lors de l'appel d'un sous-programme, l'adresse de l'instruction suivant le code d'insruction **SBR** est d'abord stockée dans la pile d'adresses. La pile d'adresses a une capacité limitée à 7 sous-programmes.
Le sous-programme se termine par l'instruction **INV SBR** qui assure le retour du sous-programme vers le programme appelant grâce à l'adresse d'origine qui est reprise de la pile d'adresses.
Si le sous-programme a été lancé à partir d'un autre espace de programme, le contrôle revient à l'espace de programme d'origine. Si le sous-programme a été démarré à partir du clavier, l'exécution s'arrête.

65 [-] - - Soustraction

Le signe **-** permet de soustraire l'opérande saisi après le signe de l'opérande saisi avant le signe.
S'il s'agit du niveau le plus bas de l'expression, le calcul peut être répété pour un autre premier opérande en appuyant plusieurs fois sur **=**.
La saisie du préfixe **INV** avant le signe **-** désactive le mode débogage (affichage de la mantisse du nombre). Le mode débogage est activé par l'instruction **INV +**.

66 [2nd] [SBR] x=t - Test égalité (EQ)

L'instruction **x=t** permet de comparer le registre **X** (contenu d'affichage) avec le registre auxiliaire **T** (chargé par la touche **x<>t** , correspond au registre **R7**). Si les registres correspondent, l'instruction suivante est exécutée. Sinon, le programme saute à l'instruction d'après.
Si le préfixe **INV** est spécifié avant le code **x=t**, la fonction inverse est exécutée - l'exécution de la commande suivante si le registre **X** est différent du registre **T**.

67 [2nd] [4] Inc* - Incrémente/décrémente un registre

L'instruction **Inc*** permet d' incrémente/décrémente le contenu d'un registre de données de la même manière que l'instruction **Inc**, seulement au lieu du paramètre d'instruction précisant le numéro de registre, ce numéro de registre est contenu dans le registre **R8**.
En plaçant le préfixe **INV** avant l'instruction **Inc***, au lieu de l'incrément de 1 du registre visé, le décrément de -1 est effectué.

68 [2nd] [5] Exc* - Échange indirect X avec un registre

L'instruction **Exc*** est utilisée pour échanger le nombre affiché avec le contenu d'un registre de données de manière similaire à l'instruction **Exc**.
Par contre, au lieu d'utiliser un paramètre d'instruction comme numéro du registre contenant la valeur à échanger, le numéro de ce registre est contenu dans le registre **R8**.
Tous les registres de données **R0** à **R79** peuvent être adressés indirectement de cette manière.
Si le préfixe **INV** est donné avant l'instruction **Exc***, le numéro de registre est extrait du registre de données **R9** (index alternatif).

69 [2nd] [6] Prd* - Produit indirect dans registre

L'instruction **Prd*** est utilisée pour multiplier ou diviser (avec le préfixe **INV**) le contenu d'un registre de données par le nombre saisi de la même manière que l'instruction **Prd**.
Par contre, au lieu d'utiliser un paramètre d'instruction comme numéro du registre contenant la valeur à multiplier, le numéro de ce registre est contenu dans le registre **R8**.
Tous les registres de **R0** à **R79** peuvent être adressés indirectement de cette manière.

70 [2nd] [+] Grd - Grades

La touche **Grd** commute les calculs des fonctions trigonométriques en grades (un angle plein est de 400 gon).

71 [RST] RST - Réinitialisation pointeur de pas

La touche **RST** permet de réinitialiser le pointeur de programme, c'est-à-dire de remettre le pointeur sur 0 (pas 00). La zone de programme sélectionnée reste inchangée.

75 [+] + - Addition

Le signe **+** permet d'ajouter l'opérande saisi après le signe à l'opérande saisi avant le signe.
S'il s'agit du niveau le plus bas de l'expression, le calcul peut être répété pour un autre premier opérande en appuyant plusieurs fois sur **=**.
La saisie du préfixe **INV** avant le signe **+** active le mode débogage (affichage de la mantisse du nombre). Le mode débogage peut être désactivé par l'instruction **INV -**.

76 [2nd] [RST] x>=t - Supérieur ou égal (GE)

L'instruction **x>=t** permet de comparer le registre **X** (contenu d'affichage) avec le registre auxiliaire **T** (chargé par la touche **x<>t**, correspond au registre **R7**). Si le registre **X** est supérieur ou égal au registre **T**, l'instruction suivant l'instruction **x>=t** est exécutée. Sinon, la commande suivante est ignorée.
Si le préfixe **INV** est spécifié avant le code **x>=t**, la fonction inverse est exécutée - l'exécution de la commande suivante si le registre **X** est plus petit que le registre **T**.

77 [2nd] [1] Inc. - Incrémente/décrémente un registre

L'instruction **Inc** incrémente (augmente de 1) le contenu d'un registre de données **R0** à **R9**, dont le numéro 0 à 9 est donné en paramètre de l'instruction. Si le préfixe **INV** est donné avant l'instruction **Inc**, l'opération inverse est effectuée - décrémentation du registre (diminution de 1).

78 [2nd] [2] Pgm - Sélection de l'espace programme

Une zone de programme peut être sélectionnée avec la touche **Pgm**. La zone de programme est sélectionnée par la saisie du numéro 0 à 9 en paramètre de l'instruction **Pgm**.
Il y a un total de 10 zones de programme indépendantes disponibles dans la calculatrice, marquées par des chiffres de 0 à 9.
Chaque zone de programme a 50 pas de programme, c'est-à-dire un total de 500 pas de programme en tout.
Lors de la commutation de la zone de programme en mode exécution (à partir du clavier), le pointeur de programme est simultanément réinitialisé au début de la zone de programme nouvellement sélectionnée (pas 00).
L'entrée de la commande **Pgm** dans un programme ne change pas l'espace du programme de façon permanente, mais seulement temporairement pour une instruction **GTO** ou **SBR** ultérieure. De cette manière, des sous-programmes peuvent être appelés ou des sauts entre les zones de programme peuvent être effectués.
Les instructions **GTO** et **SBR** n'ont pas à suivre immédiatement l'instruction **Pgm**.

79 [2nd] [3] Rand - Générateur de nombres aléatoires

La fonction **Rand** calcule un nombre aléatoire supérieur ou égal à 0 et inférieur à 1.
Le générateur **LCG** (*Linear Congruential Generator*) et la formule

Rand = (Seed - (Seed * 214013 + 2531011) mod 4294967296) / 4294967296

sont utilisés pour calculer le nombre aléatoire.
La graine du générateur a une plage de 32 bits. Le nombre généré est converti en flottant en divisant par 2^32. Cela garantit que le nombre aléatoire résultant est compris entre 0 et 1, y compris zéro mais excluant la valeur 1.
Le générateur de nombres aléatoires continue de compter chaque fois qu'il est utilisé, ce qui garantit que la séquence de nombres générés ne se répète pas (plus précisément - l'intervalle de répétition est très grand, 2^32 nombres). Chaque fois que la calculatrice est allumée, la graine du générateur est lue à partir de l'EEPROM et une nouvelle valeur est enregistrée. Cela garantit que les séquences générées ne se répètent pas après la mise sous tension de la calculatrice.

Exemple :

[RST] [LRN]		... activation du mode de programmation
00	[Lbl] [1]	... Label de début de programme (lancer le dé)
01	[(] [6] [×] [Rand] [+] [1] [)]	... génère un nombre entre 1 et 6
08	[Int]	... partie entière du nombre
09	[INV] [SBR]	... fin de programme
[LRN]		... sortie du mode de programmation
[SBR] [1]		... test, lance le dé

Le test peut être complété d'un test de comptage du nombre de sorties de chaque numéro (1 à 6) et du comptage de nombre de lancés à la seconde.

Suite Exemple :

[LRN]		... activation du mode de programmation
10	[Lbl] [2]	... Label de début de programme (nombre de lancés cumulés)
11	[SBR] [1]	... appelle le programme de lancé de dé
12	[STO] [8]	... stocke le numéro de dé dans **R8** pour utilisation comme index
13	[Inc*]	... incrément du registre indexé par le numéro de dé (**R8**)
14	[GTO] [2]	... boucle pour continuer à compter
[LRN]		... sortie du mode de programmation
[INV] [C.t]		... initialisation de tous les registres de données
[SBR] [2]		... démarrage du test de comptage
[R/S]		... attendre plusieurs minutes pour arrêter le programme
[RCL] [0] ... [RCL] [7]		vérifier lle contenu des registres **R0** à **R7**

Après 10 minutes de test, les registres **R1** à **R6** contiennent des nombres à peu près similaires (nombres de cas similaire), les registres **R0** et **R7** doivent contenir 0.
(*par exemple : 0, 2762, 2727, 2834, 2701, 2671, 2763, 0, ce qui correspond à l'hypothèse d'une vitesse de génération est de 27 lancers de dés par seconde et d'une répartition assez homogène entre 1 et 6.*)

80 [2nd] [=] Var - Dispersion

L'instruction **Var** calcule la variance des valeurs '*x*' et '*y*' saisies à l'aide de la fonction statistique **Stat**. Appuyer sur **Var** permet d'afficher la variance des valeurs '*y*', et utiliser **INV** avant l'instruction **Var** permet d'afficher la variance des valeurs '*x*'.
La variance est calculée par la formule :

var = sum(y^2)/N - (sum(y)/N)^2

La racine carrée de la variance donne l'écart type '*s*'.

81 [R/S] R/S - Démarrage/arrêt du programme

La touche **R/S** peut être utilisé pour démarrer ou arrêter un programme en cours.
Au démarrage, le programme commence à s'exécuter à partir du pointeur de programme en cours (l'adresse en cours peut être trouvée en passant au mode de programmation **LRN**).
Après l'arrêt, le programme peut ne pas toujours être en mesure de continuer à s'exécuter - l'adresse de retour du sous-programme pouvant être perdue.

83 [.] . - Séparateur décimal

Le point (**.**) est le séparateur des chiffres entiers et des chiffres décimaux d'un nombre.
En notation scientifique, si cette touche est utilisée lors de la saisie de l'exposant d'un nombre, l'édition revient à la saisie de la mantisse du nombre.

84 [+/-] +/- - Changement de signe

La touche **+/-** change le signe du nombre sur l'affichage.
Son utilisation pendant la saisie de l'exposant d'un nombre en notation scientifique, change le signe de cet exposant.

85 [=] = - Résultat du calcul

Le signe **=** est utilisé pour fermer les opérations arithmétiques ouvertes et pour exécuter les calculs.
En appuyant plusieurs fois sur la touche **=**, la dernière opération effectuée du niveau le plus bas peut être répétée. Le premier opérande est le nombre affiché, le deuxième opérande est le nombre entré pendant l'opération comme deuxième opérande (ou deuxième résultat du calcul intermédiaire).
Avertissement : Certains programmes **TI-57** originaux, pour lesquels les opérations répétitives ne sont pas autorisées, utilisent la touche **=** plus d'une fois pour entraîner une erreur. Lors de l'importation d'un programme, il peut être nécessaire de vérifier et de résoudre ce cas.

Exemple:

5 x **6** =	[30]	... 5 x 6 = 30
7 =	[42]	... 7 x 6 = 42 le deuxième opérande (**6**) est réutilisé
9 : **3** =	[3]	... 9 / 3 = 3
12=	[4]	... 12 / 3 = 4 le deuxième opérande (**3**) est réutilisé

86 [2nd] [R/S] Lbl - Label (étiquette)

L'instruction **Lbl** peut être utilisée pour marquer une début de séquence dans le programme comme une étiquette.
Dans chaque zone de programme, 10 étiquettes peuvent être utilisées, notées **Lbl** 0 à **Lbl** 9. Le numéro de l'étiquette est spécifié comme paramètre numérique 0 à 9 de l'instruction **Lbl**.
Vous pouvez sauter à l'endroit du programme marqué d'une étiquette en utilisant l'instruction de saut **GTO** ou l'Instruction d'appel de sous-programme **SBR**.
Avec l'utilisation de l'instruction de sélection d'espace de programme Pgm, des sauts et des appels de sous-programme peuvent également être effectués entre les espaces de programme.

87 [2nd] [0] x! - Factorielle

La touche **x!** est utilisé pour calculer la factorielle. La factorielle d'un nombre s'obtient en multipliant les valeurs successives 1, 2, 3, ... jusqu'au nombre spécifié. Le nombre saisi est un nombre entier compris entre 1 et 69.
(69! =
171122452428141311372468338881272839092270544893520369393648040923257279754140647424000000000000000 [99 chiffres])

Exemple :

6 x! ... factorielle du nombre 6 [6! = 1*2*3*4*5*6 = 720]

88 [2nd] [.] Stat - Statistiques

L'instruction **Stat** est utilisée pour saisir des données utilisables pour des calculs statistiques (moyenne, variance).
L'instruction utilise les registres de données **R0** à **R5** pour stocker le calcul intermédiaire.
Avant utilisation, il est conseillé de réinitialiser au préalable les registres avec la commande **INV C.t**, qui réinitialise tous les registres de données **R0** à **R79**.

Utilisation des registres de données :

- **R0** ... nombre d'éléments N
- **R1** ... la somme de *y*
- **R2** ... la somme de *y*^2
- **R3** ... la somme de *x*
- **R4** ... la somme de *x*^2
- **R5** ... la somme de *x***y*

et **R7** ... registre **T**

Lors de la saisie de données statistiques par paires (*x*, *y*), la valeur '*x*' est saisie en premier et en appuyant sur **x<>t** elle est transférée au registre **T** (registre **R7**).
Ensuite, la valeur '*y*' est saisie et en appuyant sur Stat, les valeurs '*x*' et '*y*' sont enregistrées.
L'affichage (dans le registre **X**) indiquera le nombre d'éléments 'n' insérés jusqu'à présent. Le contenu du registre **T** (la valeur de '*x*') est augmenté de 1 avec l'instruction Stat, car si les valeurs de '*x*' doivent s'incrémenter de 1, il n'est pas nécessaire de les insérer, il suffit d'insérer la valeur initiale de '*x*' dans le registre **T**, puis écrivez uniquement les valeurs '*y*'. S'il n'est pas nécessaire d'évaluer des paires de valeurs (*x*, *y*), entrez simplement la valeur '*y*'.
Si le préfixe **INV** est utilisé avant l'instruction Stat, la valeur saisie sera soustraite. C'est ainsi que vous pouvez corriger une valeur erronée - entrez la valeur '*x*' des données erronées, appuyez sur **x<>t**, entrez la valeur '*y*' des données erronées et appuyez sur **INV Stat** pour supprimer les données erronées. Le contenu du registre **T** (avec la valeur '*x*') est alors décrémenté de 1.
Ensuite, la saisie des nouvelles données correctes peuvent être poursuivies. S'il n'est pas nécessaire d'évaluer des paires de valeurs (*x*, *y*), il suffit de saisir uniquement la valeur '*y*' de la donnée erronée.

Exemple :

INV	C.t			... initialisation des registres de données
9	6	Stat	[1]	... 1ère donnée 96
8	1	Stat	[2]	... 2ème donnée 81
9	7	Stat	[3]	... 3ème donnée erronée
9	7	Stat	[2]	... annulation de la 3ème donnée
8	7	Stat	[3]	... 3ème donnée corrigée 87
7	0	Stat	[4]	... 4ème donnée 70
9	3	Stat	[5]	... 5ème donnée 93
7	7	Stat	[6]	... 6ème donnée 77
Mean			[84]	... moyenne des valeurs saisies
Var			[81,333...]	... écart
√x			[9.0185...]	... écart-type
RCL	1		[504]	... somme de toutes les valeurs

89 2nd +/- Mean - Moyenne

L'instruction **Mean** calcule la moyenne des valeurs '*x*' et '*y*' saisies à l'aide de la fonction statistique **Stat**.
Après avoir appuyé sur **Mean**, l'écran affiche la moyenne des valeurs '*y*'.
La saisie du préfixe **INV** avant l'instruction **Mean** calcule la moyenne des valeurs '*x*'.

15. Exemples de programmes

1. Lancez les dés (version ET-57)

Le programme utilise le générateur de nombre aléatoire interne.

Utilisation :

SBR 1	... génère un nombre entier aléatoire compris entre 1 et 6
R/S	... nombre suivant

Programme :

00	86 1	Lbl 1	étiquette de début de sous-programme
01	43	(	calcule de (6 x Rand + 1), nombre aléatoire 1..6.9999
02	6	6	
03	55	x	
04	79	Rand	nombre aléatoire de 0 à 0.99999...
05	75	+	
06	1	1	
07	44	)	
08	49	Int	partie entière du nombre
09	-61	INV SBR	fin de sous-programme
10	51 1	GTO 1	numéro suivant avec R/S

2. Lancez les dés (version TI-57)

La **TI-57** d'origine ne possédant pas de fonction **Rand** (générateur de nombre aléatoire) le programme de lancé de dé de cette dernière coûte le double de nombre de pas plus l'utilisation d'un registre.
(ce programme est aussi applicable, avec quelques variantes mineures, aux modèles successeurs de la **TI-57**, soit les **TI-57 LCD**, **TI-57 II** et **TI-62**)

Registre :

R0 ... Semence du générateur de nombre aléatoire (Seed)

Utilisation :

x [STO] [1] ... initialiser le générateur de nombre avec une semence = x

[SBR] [1] ... génère un nombre entier aléatoire compris entre 1 et 6

[R/S] ... nombre suivant

Programme :

00	86 1	Lbl 1	étiquette de début de sous-programme
01	43	(	
02	43	(	\
03	33 0	RCL 0	| Rappel de la semence (Seed)
04	75	+	|
05	30	pi	|
06	44	)	|
07	35	y^x	\> Calcul de frac((Seed + pi)^8)
08	8	8	|
09	44	)	|
10	43	(	|
11	-49	INV Int	/
12	32 0	STO 0	Stocke la nouvelle semence (Seed)
13	55	x	\
14	6	6	|
15	75	+	\> Valeur du dé de 1 à 6
16	1	1	|
17	44	)	/
18	49	Int	
19	-61	INV SBR	
20	51 1	GTO 1	numéro suivant avec R/S

3. Serpent lumineux avec des LED

Le programme contrôle un appareil externe - le cadre d'effet ERAM100. En écrivant un octet sur le port 0, 8 LED sont contrôlées. L'octet écrit est une combinaison de bits éclairant des LED individuelles, comme la somme des poids des bits :

- bit 0 ... poids 1 ... LED 1
- bit 1 ... poids 2 ... LED 2
- bit 2 ... poids 4 ... LED 3
- bit 3 ... poids 8 ... LED 4
- bit 4 ... poids 16 ... LED 5
- bit 5 ... poids 32 ... LED 6
- bit 6 ... poids 64 ... LED 7
- bit 7 ... poids 128 ... LED 8

Utilisation :

[RST] [R/S] ... démarre le serpent :
3 LED lumineuses circulent le long du bandeau de LED

Registres :

R5 ... sauvegarde position du serpent
R8 ... index, -1 est l'index du port 0 (sortie vers LED)

Programme :

00	7	7	Valeur des bits d'apparition du serpent (LED 1,2,3 s'allume)
01	32 5	STO 5	Stocke la position du serpent
02	1	1	
03	84	+/-	
04	32 8	STO 8	définit l'index de R8 sur -1, qui est l'adresse du port 0
05	2	2	
06	5	5	
07	6	6	
08	22	x<>t	stocke 256 dans T pour le test de débordement de modèle
09	86 9	Lbl 9	étiquette du début du cycle
10	33 5	RCL 5	rappelle la position du serpent
11	55	x	
12	2	2	
13	85	=	(position x 2) correspond à un décalage de 1 bit vers la gauche
14	-76	INV x>=t	position < 256 ?
15	51 8	GTO 8	pas de débordement de position, passez à Lbl 8
16	65	-	
17	2	2	
18	5	5	
19	5	5	
20	85	=	rotation : - 256 (enlever le bit 7) + 1 (ajouter le bit 0)
21	86 8	Lbl 8	
22	32 5	STO 5	stocke la nouvelle position du serpent
23	57	STO*	le stockage à l'adresse -1 envoie l'octet au port 0
24	36	Pause	petite pause de 0,25 seconde
25	51 9	GTO 9	répétition de la boucle

4. Calcul du polynôme

Le programme calcule la valeur du polynôme

P(x) = a0 + a1*x + a2*x^2 + ... an*x^n

d'ordre n pour le x donné, si les coefficients a0 à an sont entrés.

Registres :

R0 ... compteur d'itération Dsz
R6 ... nombre de coefficients N (= ordre du polynôme n + 1)
R8 ... registre d'index
R9 ... entrée x
R10 et suivants ... coefficients polynomiaux a0, a1, ... an (N coefficients)

Utilisation :

Exemple, polynôme P(x) = 2 - 3x + x^2 :

RST R/S	... commencer à saisir les coefficients
2 R/S	... saisir le coefficient a0 = 2
3 +/- R/S	... entrée du coefficient a1 = -3
1 R/S	... entrée du coefficient a2 = 1
2 SBR 1	... calcul de la valeur du polynôme P(2) = 0
1 +/- R/S	... calcul de la valeur du polynôme P(-1) = 6
1 5 R/S	... calcul de la valeur du polynôme P(15) = 182

Programme :

00	15	CLR	remise à zéro
01	32 6	STO 6	mise à zéro du nombre de coefficients N
02	9	9	indice du premier coefficient 10 - 1 = 9
03	32 8	STO 8	paramètres de registre d'index
04	86 9	Lbl 9	étiquettes de boucle pour la saisie de coefficients
05	81	R/S	arrêt du programme, attente du coefficient suivant
06	77 8	Inc 8	incrémentation du registre d'index **R8**
07	57	STO*	stocker le coefficient
08	77 6	Inc 6	incrémentant le nombre de coefficients N
09	51 9	GTO 9	continuer avec le coefficient suivant
10	86 8	Lbl 8	étiquette de répétition de calcul
11	81	R/S	afficher le résultat et attendre un autre 'x'
12	86 1	Lbl 1	l'étiquette de début de la fonction de quantification polynomiale
13	32 9	STO 9	enregistre 'x' saisi
14	43	(	
15	33 6	RCL 6	nombre de coefficients N
16	32 0	STO 0	préparation du compteur **Dsz**
17	75	+	
18	9	9	N + 9 = indice du dernier coefficient an
19	44	)	
20	32 8	STO 8	réglage du registre d'index **R8**
21	58	RCL*	rappel le dernier coefficient an
22	86 7	Lbl 7	étiquette de début de boucle
23	-56	INV Dsz	décrémenter **R0** et sauter l'instruction quand > 0
24	51 8	GTO 8	boucle si **R0** diffère de 0 sinon fin de boucle
25	-77 8	INV Inc 8	décrémente le registre d'index **R8**
#	43	(	
27	14	CE	Restaure dernier registre **X**
28	55	x	
29	33 9	RCL 9	Calcul Affichage * 'x' saisi
30	75	+	
31	58	RCL*	ajout d'un autre coefficient ai (issu de l'indice **R8**)
32	44	)	
33	51 7	GTO 7	continue la boucle

5. Nombres complexes

Arithmétique des nombres complexes.
Lorsque vous saisissez un nombre, saisissez la partie réelle, appuyez sur x<>t et saisissez la partie imaginaire.
Dans le registre T (registre R7) il y aura la partie réelle du nombre, et dans le registre X (affichage) la partie imaginaire.
Lors de la lecture du résultat, après avoir appuyé sur x<>t vous lirez la partie réelle, après avoir appuyé une seconde fois sur x<>t vous lirez la partie imaginaire.

Pour les fonctions à 2 opérandes, spécifiez le premier opérande (**X**) à l'aide de **SBR 1**.
Spécifiez ensuite le deuxième opérande (**Y**) et appelez la fonction.
Le résultat devient en même temps le nouveau premier opérande (**X**).
Pour les fonctions à 1 opérande, spécifiez l'opérande (**Y**) et appelez la fonction.
Le premier opérande d'origine (**X**) reste inchangé.

Le programme occupe 2 espaces de programme, **Pgm 0** et **Pgm 1**.

Remarque : Les fonctions ln(Y) et exp(Y) changent la mesure angulaire en radians.

Utilisation :
Espace programme **Pgm 0** :

SBR 1	... entrée du premier opérande X
SBR 2	... ajout du deuxième opérande X+Y
SBR 3	... soustraction du deuxième opérande X-Y
SBR 4	... multiplication par le deuxième opérande X*Y
SBR 5	... division par le deuxième opérande X/Y
SBR 6	... négation -Y

Remarque : La fonction **SBR 5** appelle la fonction **SBR 1** à partir de **Pgm 1**.

Espace programme **Pgm 1** :

SBR 1	... inverse de 1/Y
SBR 2	... carré de Y^2
SBR 3	... racine carrée de sqrt(Y)
SBR 4	... logarithme népérien de ln(Y)
SBR 5	... exposant naturel exp(Y)

Exemple : calcul de exp(((2+3i) - (1-i)) / (4+5i))

Pgm 0		... sélection de l'espace programme 0
2 x<>t 3 SBR 1		... entrée du premier opérande (2+3i)
1 x<>t 1 +/- SBR 3		... soustraction du deuxième opérande (1-i)
x<>t	[1]	
x<>t	[4]	... résultat intermédiaire = (1+4i)
4 x<>t 5 SBR 5		... division par opérande (4+5i)
x<>t	[0.5853...]	
x<>t	[0.2682...]	... résultat intermédiaire = (0.5853...+0.2682...i)
Pgm 1		... sélection de l'espace programme 1
1 SBR 5		... exposant naturel exp (Y)
x<>t	[1.7314...]	
x<>t	[0.4760...]	... résultat (1.7314... + 0.4760..i)

Registres :

R1 ... 'a', la partie réelle du premier opérande de X
R2 ... 'b', la partie imaginaire du premier opérande de X
R3 ... 'c', la partie réelle du deuxième opérande de Y
R4 ... 'd', la partie imaginaire du deuxième opérande Y
R7 ... registre **T**

Programme :

Pgm 0 ... sélection de l'espace programme 0

- calcul de la soustraction du deuxième opérande X-Y

00	86 3	Lbl 3	étiquette fonction X-Y
01	61 6	SBR 6	appel la fonction de négation -Y

- calcul du second opérande X+Y

X + Y = (a + c) + (b + d)*i

02	86 2	Lbl 2	étiquette fonction X+Y
03	34 2	SUM 2	ajouter la partie imaginaire de Y à X
04	33 2	RCL 2	rappel de la partie imaginaire de X
05	22	x<>t	échange de parties réel <> imaginaire
06	34 1	SUM 1	ajouter la partie réelle de Y à X
07	33 1	RCL 1	rappel de la partie réelle de X
08	22	x<>t	échange de parties réel <> imaginaire

- saisie du premier opérande X

09	86 1	Lbl 1	étiquette fonction X
10	32 2	STO 2	stocker la partie imaginaire de X
11	22	x<>t	échange de parties réel <> imaginaire
12	32 1	STO 1	stocker la partie réelle de X
13	22	x<>t	échange de parties réel <> imaginaire
14	-61	INV SBR	

- calcul de la division par le second opérande X/Y

15	86 5	Lbl 5	étiquette fonction X/Y
16	78 1	Pgm 1	choix de l'espace de programme 1
17	61 1	SBR 1	appel de fonction inverse de 1/Y

- calcul de la multiplication par le second opérande X*Y

X * Y = (a*c - b*d) + (a*d + b*c)*i

18	86 4	Lbl 4	étiquette fonction X*Y
19	32 4	STO 4	stockage de la partie imaginaire Y (=d)
20	22	x<>t	échange de parties réel <> imaginaire
21	43	(	
22	32 3	STO 3	stockage de la partie réelle de Y (=c)
23	55	x	
24	33 1	RCL 1	partie réelle de X (=a)
25	65	-	
26	33 2	RCL 2	rappel de la partie imaginaire de X (=b)
27	55	x	
28	33 4	RCL 4	rappel de la partie imaginaire de Y (=d)
29	44	)	calcul (a*c - b*d) partie réelle
30	22	x<>t	échange de parties réel <> imaginaire
31	43	(	
32	33 1	RCL 1	rappel de la partie réelle de X (=a)
33	55	x	
34	33 4	RCL 4	rappel de la partie imaginaire de Y (=d)
35	75	+	
36	33 2	RCL 2	rappel de la partie imaginaire X (=b)
37	55	x	
38	33 3	RCL 3	rappel de la partie réelle de Y (=c)
39	44	)	calcul (a*d + b*c) partie imaginaire
40	51 1	GTO 1	stockage du résultat dans X

- Changement signe de l'opérande -Y

41	86 6	Lbl 6	étiquette fonction -Y
42	84	+/-	négation de la partie imaginaire
43	22	x<>t	échange de parties réel <> imaginaire
44	84	+/-	négation de la partie réelle
45	22	x<>t	échange de parties réel <> imaginaire
46	-61	INV SBR	

Pgm 1 ... sélection de l'espace programme 1

- inverse de l'opérande 1/Y

1/Y = (c - d*i)/(c^2 + d^2)

00	86 1	Lbl 1	étiquette de la fonction 1/Y
01	43	(	
02	84	+/-	change signe partie imaginaire de Y (=d)
03	45	:	
04	43	(	
05	23	x^2	j^2
06	75	+	
07	22	x<>t	
08	32 3	STO 3	stockage de la partie réelle de Y (=c)
09	23	x^2	c^2
10	44	)	calcul (c^2 + d^2)
11	-39 3	INV Prd 3	division c par (c^2 + d^2)
12	44	)	fin calcul de la partie imaginaire
13	22	x<>t	échange de parties réel <> imaginaire
14	33 3	RCL 3	rappel de la nouvelle partie réelle
15	22	x<>t	échange de parties réel <> imaginaire
16	-61	INV SBR	

- élévation au carré opérande Y^2

17	86 2	Lbl 2	étiquette de la fonction Y^2
18	43	(	
19	-27	INV P->R	convertir en coordonnées polaires
20	55	x	l'angle sera multiplié par *2
21	22	x<>t	échange de parties réel <> imaginaire
22	23	x^2	élévation au carré rayon
23	86 9	Lbl 9	
24	22	x<>t	échange de parties réel <> imaginaire
25	2	2	
26	44	)	angle * 2 (ou angle / 2) est calculé ici
27	27	P->R	convertir en coordonnées cartésiennes
28	-61	INV SBR	

- racine carrée opérande sqrt(Y)

29	86 2	Lbl 3	étiquette de la fonction sqrt(Y)
30	43	(	
31	-27	INV P->R	conversion en coordonnées polaires
32	45	:	l'angle sera divisible par /2
33	22	x<>t	échange de parties réel <> imaginaire
34	24	Vx	racine carrée du rayon
35	51 9	GTO 9	réalisation de la fonction

- logarithme népérien de l'opérande ln(Y)

36	86 4	Lbl 4	étiquette de la fonction ln(Y)
37	60	Rad	passer aux radians
38	-27	INV P->R	convertit en coordonnées polaires
39	22	x<>t	échange de parties réel <> imaginaire
40	13	lnx	logarithme du rayon
41	22	x<>t	échange de parties réel <> imaginaire
42	-61	INV SBR	

- exposant naturel de l'opérande exp(Y)

43	86 5	Lbl 5	étiquette fonction exp(Y)
44	22	x<>t	échange de parties réel <> imaginaire
45	-13	INV lnx	exposant exp(x)
46	22	x<>t	échange de parties réel <> imaginaire
47	60	Rad	passer aux radians
48	27	P->R	convertit en coordonnées cartésiennes
49	-61	INV SBR	

6. Approximation de Ramanujan de la factorielle x !

Calcul de la factorielle (y compris les nombres décimaux) à l'aide de l'approximation de Ramanujan.
Précision atteinte : Valeurs autour de 1 précision 3 chiffres, valeurs autour de 69 (maximum) précision 10 chiffres.

Formule : x ! = sqrt(pi) * (x/e)^x * (((8*x + 4)*x + 1)*x + 1/30)^(1/6).

Utilisation :

x [SBR] [1] ... calcule de la factorielle x!

x' [R/S] ... calcul nombre suivant

Exemples :

[1] [SBR] [1] ... 1! = 1,00028..., la valeur correcte doit être 1,0

[1] [.] [2] [R/S] ... 1.2 ! = 1,101987..., doit être 1,101802...

[1] [0] [R/S] ... 10 ! = 3628800.3116, doit être 3628800

[6] [9] [R/S] ... 69 ! = 1,7112245244+98, doit être 1,7112245243+98

Registres :

R1 ... valeur saisie de x

Programme :

00	86 1	Lbl 1	début de programme
01	43	(	
02	43	(	
03	32 1	STO 1	stockage de la valeur saisie
04	45	:	
05	1	1	
06	-13	INV lnx	calcul de la constante e
07	44	)	calcul de x/e
08	35	y^x	
09	33 1	RCL 1	calcul de (x/e)^x
10	55	x	
11	30	pi	
12	24	Vx	calcul de racine carrée de pi
13	55	x	
14	43	(	
15	43	(	
16	43	(	
17	8	8	
18	55	x	
19	33 1	RCL 1	calcul de 8 * x
20	75	+	
21	4	4	
22	44	)	calcul de (8*x + 4)
23	55	x	
24	33 1	RCL 1	
25	75	+	
26	1	1	
27	44	)	calcul de ((8*x + 4)*x + 1)
28	55	x	
29	33 1	RCL1	
30	75	+	
31	3	3	
32	0	0	
33	25	1/x	
34	44	)	calcul de ...*x + 1/30)
35	-35	INV y^x	racine 6ème
36	6	6	
37	44	)	
38	-61	INV SBR	
39	51 1	GTO 1	répéter le calcul pour une autre valeur

7. Approximation de Stirling de la factorielle ln(x !)

Calcul de la factorielle (y compris les nombres décimaux) à l'aide de l'approximation de Stirling.
Précision atteinte : Valeurs autour de 1 précision 3 chiffres, valeurs autour de 69 (maximum x !) précision 10 chiffres, valeurs autour de 200 précision à 15 chiffres.

Formule : ln(x!) = x*ln(x)+ln(sqrt(2*pi))-x+ln(sqrt(x+1/6+1/72/x-31/6480/x^2)).

Utilisation :

x [SBR] [1] ... calcule du logarithme de la factorielle ln(x!)

x' [R/S] ... calcul nombre suivant

x [SBR] [2] ... calcule la factorielle x!

x' [R/S] ... calcul nombre suivant

Exemples :

[1] [SBR] [2] ... 1! = 0.99990..., la valeur correcte doit être 1,0

[1] [.] [2] [R/S] ... 1.2 ! = 1.10177..., doit être 1,101802...

[1] [0] [R/S] ... 10 ! = 3628800.1322, doit être 3628800

[6] [9] [R/S] ... 69 ! = 1.7112245243+98, valeur correcte

[1] [0] [0] [0] [SBR] [1] ... ln(1000!) = 5912.1281785, valeur correcte

Registres :

R1 ... valeur saisie de x

Programme :

- calcul de ln(x!)

0	81 1	Lbl 1	début de programme ln(x!)
1	43	(	
2	32 1	STO 1	stockage de la valeur saisie de x
3	55	x	
4	13	lnx	calcul de x*ln(x)
5	75	+	
6	43	(	
7	2	2	
8	55	x	
9	30	pi	
10	44	)	
11	24	Vx	
12	13	lnx	calcul de ln(sqrt(2*pi))
13	65	-	
14	33 1	RCL 1	Rappel x
15	75	+	
16	43	(	
17	33 1	RCL 1	Rappel x
18	75	+	
19	6	6	
20	25	1/x	nombre + 1/6
21	75	+	
22	7	7	
23	2	2	
24	25	1/x	
25	45	:	
26	33 1	RCL 1	nombre + 1/72/x
27	65	-	
28	3	3	
29	1	1	
30	45	:	
31	6	6	
32	4	4	
33	8	8	
34	0	0	
35	45	:	
36	33 1	RCL 1	
37	23	x^2	nombre - 31/6480/x^2
38	44	)	
39	24	Vx	racine carrée du total
40	13	lnx	
41	44	)	
42	-61	INV SBR	
43	51 1	GTO 1	répéter le calcul pour une autre valeur

- calcul de x! [soit e^ln(x!)]

44	86 2	Lbl 2	
45	61 1	SBR 1	appel de la fonction ln(x!)
46	-13	INV lnx	exposant, x! = e^ln(x!)
47	-61	INV SBR	
48	51 2	GTO 2	répéter le calcul pour une autre valeur

8. Détermination des zéros d'une fonction

Le programme recherche les passages par zéro de la fonction utilisateur. Le programme occupe les espaces de programme **Pgm 0** et **Pgm 1**. La fonction utilisateur est entrée dans l'espace de programme **Pgm 2** et marquée avec l'étiquette **Lbl 0**. La fonction peut utiliser temporairement le registre **R7** (registre **T**) et les registres **R10** et supérieurs.

La fonction recherche d'abord un intervalle de pas dx = (xmax - xmin)/10 dans lequel le signe de y change. Il affine alors le lieu de passage en divisant l'intervalle par zéro jusqu'à l'écart eps = dx/100000. La taille du pas dx peut être fixée à l'adresse 22. Avec une grande valeur du pas dx, certains passages à zéro peuvent être sautés, une petite valeur de dx ralentit la recherche. L'amplitude de l'écart eps peut être définie à l'adresse 27. La valeur eps affecte la précision de recherche de passage par zéro réalisable au prix d'un ralentissement de la recherche.

Utilisation :

SBR 0	... fonction utilisateur : calcul de la valeur y pour le x donné
SBR 1	... saisie de la limite inférieure de xmin
SBR 2	... saisie de la limite supérieure de xmax
SBR 3	... déterminer le premier passage par zéro
SBR 4	... déterminer le prochain passage par zéro

Exemples :

zéros de la fonction f(x) = 4*sin(x) + 1 - x

*Le programme de cette fonction f(x) doit être créé dans l'espace **Pgm 2** (voir pages suivantes).*

Pgm 0	... sélection de l'espace programme 0
3 +/- SBR 1	... saisie de la limite inférieure xmin = -3
3 SBR 2	... saisie de la limite supérieure xmax = 3
SBR 3	... trouver le premier passage par zéro = -2.2100...
SBR 4	... trouver le deuxième passage par zéro = -0.3421...
SBR 4	... trouver le troisième passage par zéro = 2.7020...
SBR 4	... quatrième introuvable = 9.999+99

Registres :

R0 ... la valeur y de la fonction testée
R1 ... limite inférieure xmin
R2 ... limite supérieure de xmax
R3 ... intervalle delta dx
R4 ... x le début de l'intervalle
R5 ... x fin d'intervalle
R6 ... x courant
R7 ... registre T, temporairement disponible pour SBR 0
R8 ... écart eps (minimum dx)
R9 ... limite inférieure actuelle de xmin2

Programme :

Pgm 0 ... sélection de l'espace programme 0

- calcul de la valeur de la fonction utilisateur

00	86 0	Lbl 0	Appel fonction
01	78 2	Pgm 2	choix de l'espace programme 2
02	61 0	SBR 0	appel de la fonction utilisateur
03	-61	INV SBR	

- Saisie de la limite inférieure de xmin

04	86 1	Lbl 1	saisie de la limite inférieure xmin
05	32 1	STO 1	enregistre la limite inférieure de xmin
06	-61	INV SBR	

- Saisie de la limite supérieure de xmax

07	86 2	Lbl 2	saisie de la limite supérieure de xmax
08	32 2	STO 2	enregistre la limite supérieure de xmax
09	-61	INV SBR	

- Affichage erreur : suivant introuvable

10	86 9	Lbl 9	Affichage erreur
11	15	CLR	
12	25	1/x	provoque erreur 9.99999 99
13	86 8	Lbl 8	
14	-61	INV SBR	

- Routine pour chercher le premier zéro

15	86 3	Lbl 3	première recherche de zéro de la fonction
16	33 2	RCL 2	rappel limite supérieure xmax
17	65	-	
18	33 1	RCL 1	rappel limite inférieure xmin
19	32 9	STO 9	stocke limite inférieure xmin
20	85	=	calcul différence (xmax - xmin)
21	45	:	
22	1	1	**<-** taille du pas dx
23	0	0	
24	85	=	calcul dx = intervalle / 10
25	32 3	STO 3	stocke l'intervalle delta dx
26	45	:	
27	5	5	**<-** amplitude de l'écart eps
28	-18	INV log	constante 100000
29	85	=	calcul écart maximal eps
30	32 8	STO 8	stocke écart eps

- Routine pour chercher le zéro suivant

31	86 3	Lbl 4	autre recherche de zéro de la fonction
32	33 2	RCL 2	rappel limite supérieure de xmax
33	22	x<>t	xmax dans T
34	33 9	RCL 9	rappel limite inférieure xmin actualisé
35	76	x>=t	limite supérieure atteinte ?
36	51 9	GTO 9	passage suivant introuvable
37	32 4	STO 4	stocke x début d'intervalle
38	75	+	
39	33 3	RCL 3	rappel intervalle delta dx
40	85	=	
41	32 9	STO 9	stocke nouvelle limite inférieure xmin
42	32 5	STO 5	stocke x à la fin d'intervalle
43	61 0	SBR 0	calcul de y à la fin de l'intervalle
44	32 0	STO 0	stocke la valeur de y à la fin de l'intervalle
45	33 4	RCL 4	rappel x au début de l'intervalle
46	32 6	STO 6	stocke le nouveau x actualisé
47	61 0	SBR 0	calcul de y pour le nouveau x actualisé
48	78 1	Pgm 1	choix de l'espace de programme 1
49	51 9	GTO 9	suite de la fonction dans l'espace 1

Pgm 1 ... sélection de l'espace programme 1

- Poursuite de l'opération de recherche pour le zéro suivant

00	86 9	Lbl 9	suite de la recherche du zéro suivant
01	19	C.t	mise à zéro T
02	66	x=t	la valeur de y est-elle égale à 0 ?
03	51 8	GTO 8	valeur à zéro trouvé
04	55	x	
05	38 0	Exc 0	y_actuel * y_initial
06	85	=	
07	-76	INV x>=t	si multiple < 0, zéro trouvé
08	51 7	GTO 7	si zéro trouvée, affiner résultat
09	78 0	Pgm 0	passage à l'espace programme 0
10	51 4	GTO 4	continuer avec l'intervalle suivant

- boucle pour affiner le résultat

11	86 7	Lbl 7	affinage précision du zéro trouvé
12	33 4	RCL 4	x au début de l'intervalle
13	75	+	
14	33 5	RCL 5	x à la fin de l'intervalle
15	85	=	
16	45	:	
17	2	2	
18	85	=	calcul centre de l'intervalle x
19	32 6	STO 6	stocke nouveau x actualisé
20	33 8	RCL 8	rappel déviation ps
21	22	x<>t	stockage de l'écart eps dans T
22	33 5	RCL 5	rappel x à la fin de l'intervalle
23	65	-	
24	33 4	RCL 4	rappel x au début de l'intervalle
25	85	=	calcul longueur de l'intervalle
26	-76	INV x>=t	intervalle < epsilon ?
27	51 6	GTO 6	si intervalle < epsilon, zéro trouvé OK
28	33 6	RCL 6	rappel x actualisé (centre de l'intervalle)
29	78 2	Pgm 2	choix de l'espace programme 2
30	61 0	SBR 0	calcul de y pour le nouveau x
31	55	x	
32	33 7	RCL 0	rappel y précédent
33	85	=	
34	19	C.t	mise à zéro T
35	66	x=t	y = 0 ?
36	51 6	GTO 6	zéro trouvé
37	76	x>=t	si produit > 0, pas de changement de signe
38	51 5	GTO 5	pas de changement de signe, passer à x supérieur
39	33 6	RCL 6	rappel x actualisé (centre de l'intervalle)
40	32 5	STO 5	décaler vers le bas, le milieu sera la nouvelle fin
41	51 7	GTO 7	poursuite de la boucle affinage zéro
42	86 5	Lbl 5	étiquette pour passer à un x supérieur
43	33 6	RCL 6	rappel x actualisé (centre de l'intervalle)
44	32 4	STO 4	monter, le centre sera un nouveau départ
45	51 7	GTO 7	poursuite de la boucle affinage zéro
46	86 6	Lbl 6	résultat OK
47	33 6	RCL 6	rappel résultat
48	78 0	Pgm 0	choix de l'espace programme 0
49	51 8	GTO 8	retour de fonction

Pgm 2 ... sélection de l'espace programme 2

- exemple de fonction utilisateur f(x) = 4*sin(x)+1-x

00	86 0	Lbl 0	étiquette fonction utilisateur
01	43	(	
02	32 7	STO 7	stocker X initial
03	60	Rad	passer aux radians
04	28	sin	\
05	55	x	|
06	4	4	|
07	75	+	> fonction utilisateur 4*sin(x)+1-x
08	1	1	|
09	65	-	|
10	33 7	RCL 7	|
11	44	)	/
12	-61	INV SBR	

9. Intégrale de Simpson de la fonction.

Le programme calcule l'intégrale numérique de la fonction utilisateur par l'approximation de Simpson. Le programme est stocké dans l'espace programme **Pgm 0**, la fonction utilisateur est créée dans l'espace programme **Pgm 1** sous l'étiquette **Lbl 0**.
Le nombre d'étapes spécifié n doit être un nombre pair.

Utilisation :

	CLR RST	... initialiser les opérations et initialiser le pointeur de programme
xmin	R/S	... saisie de la limite inférieure
xmax	R/S	... saisie de la limite supérieure
n	R/S	... saisie du nombre d'étapes (un nombre pair !) et calcul

Exemples :

intégrale de la fonction f(x) = 1/(cos(x) + 2) dans l'intervalle 0 à pi/2

Le programme de cette fonction doit être créé dans l'espace ***Pgm 1****, sous l'étiquette Lbl 0 (voir pages suivantes).*

Pgm 0	... sélection de l'espace programme 0
CLR RST	... opérations d'initialisation et raz du pointeur (pas 00)
0 R/S	... saisie de la limite inférieure
π ÷ 2 = R/S	... saisie de la limite supérieure
2 0 R/S	... saisie du nnombre d'étapes (pair !) et calcul [0.6046..]

Résultat pour 2 étapes =	0.604998903 (précision 3 chiffres)
Résultat pour 4 étapes =	0.604619709 (précision 4 chiffres)
Résultat pour 10 étapes =	0.604600227 (précision 6 chiffres)
Résultat pour 20 étapes =	0.604599815 (précision 7 chiffres)
Résultat pour 100 étapes =	0.604599788 (précision 10 chiffres)
Résultat pour 200 étapes =	0.604599788 (précision 11 chiffres)
Résultat pour 1000 étapes =	0.604599788 (précision 13 chiffres)
Résultat pour 2000 étapes =	0.604599788 (précision 13 chiffres)
Résultat de référence =	0.604599788

Registres :

R0 ... nombre d'étapes n, compteur de boucles
R1 ... limite inférieure xmin
R2 ... incrément d'étapes dx
R3 ... perte du résultat de l'intégrale y

Programme :

Pgm 0 ... sélection de l'espace programme 0

0	32 1	STO 1	stocke la limite inférieure de xmin
1	81	R/S	attente de la limite supérieure xmax
2	65	-	
3	33 1	RCL 1	rappel limite inférieure xmin
4	85	=	calcul intervalle (xmax - xmin)
5	32 2	STO 2	stocke l'intervalle
6	81	R/S	attente du nombre d'étapes
7	32 0	STO 0	stocke nombre d'étapes n
8	-39 2	INV Prd 2	calcul de l'incrément d'étape : intervalle / n,
9	61 9	SBR 9	calcul des x et y courants
10	32 3	STO 3	stocke résultat de l'intégrale I
11	86 8	Lbl 8	début de la boucle de calcul
12	-77 0	INV Inc 0	décrémentation de l'index **R0** de la boucle
13	61 9	SBR 9	calcul des x et y courants
14	55	x	
15	4	4	
16	85	=	calcul de y * 4
17	34 3	SUM 3	
18	-56	INV Dsz	décrémente registre **R0**, sauter si > 0
19	51 7	GTO 7	aller à fin de la boucle lorsque **R0** = 0
20	61 9	SBR 9	calcul des x et y courants
21	55	x	
22	2	2	
23	85	=	calcul de y * 2
24	34 3	SUM 3	
25	51 8	GTO 8	étape suivante de la boucle
26	86 7	Lbl 7	fin de la boucle
27	61 9	SBR 9	calcul des x et y courants
28	34 3	SUM 3	
29	33 2	RCL 2	rappel intervalle dx
30	45	:	
31	3	3	calcul incrément dx / 3 * y
32	55	x	
33	33 3	RCL 3	perte de résultat y
34	85	=	
35	81	R/S	arrêt, affichage du résultat
36	86 7	Lbl 9	affiche le calcul des x et y actuels
37	33 1	RCL 1	limite inférieure xmin
38	75	+	
39	33 0	RCL 0	compteur de pas
40	55	x	
41	33 2	RCL 2	incrément d'étape dx
42	85	=	calcul coordonnée actuelle x
43	78 1	Pgm 1	choix de l'espace programme 1
44	51 0	GTO 0	aller à la fonction utilisateur

[Pgm] [1] ... sélection de l'espace programme 1

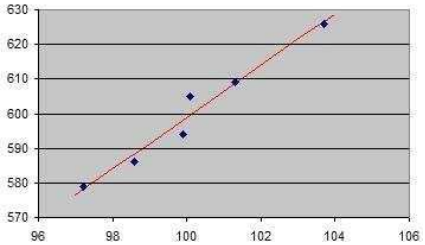
- exemple de fonction utilisateur f(x) = 1/(cos(x) + 2)

00	86 0	Lbl 0
01	60	Rad
02	29	cos
03	75	+
04	2	2
05	85	=
06	25	1/x
07	-61	INV SBR

10. Droite de régression linéaire.

Le programme calcule les coefficients de la droite de régression linéaire approximative en utilisant la méthode des moindres carrés. Le couple de valeurs (X,Y) est saisi à l'aide de la fonction statistique **Stat**. La droite de régression a la forme ***y = m*x + b***.

Le coefficient 'm', c'est-à-dire la pente de la droite, est calculé selon la formule ***m = (sum(x*y) - sum(x)*sum(y)/N) / (sum(x^2) - somme(x)^2/ N)***.
Le coefficient 'b', c'est-à-dire le déplacement de la droite dans la direction Y, est calculé selon la formule ***b = (somme(y) - m*somme(x))/N***.



Utilisation :

Saisissez les paires (X,Y) à l'aide de la fonction [Stat]

[CLR] [RST] ... initialisation du pointeur de programme (pas 00)

[R/S] ... calcul du coefficient 'm'

[R/S] ... calcul du coefficient 'b'

Les fonctions suivantes ne peuvent être appelées qu'après avoir calculé 'm' et 'b'.

x [SBR] [1] ... calcul de y pour une valeur donnée de x

y [SBR] [2] ... calcul de x pour une valeur donnée de y

Exemple :

CLR INV C.t ... effacement de tous les registres **X**, **R0** à **R79**

1 0 1 . 3 x≷t 6 0 9 Stat
... saisie point 1 (101.3, 609)

1 0 3 . 7 x≷t 6 2 6 Stat
... saisie point 2 (103.7, 626)

9 8 . 6 x≷t 5 8 6 Stat
... saisie point 3 (98,6, 586)

9 9 . 9 x≷t 5 9 4 Stat
... saisie point 4 (99.9, 594)

9 7 . 2 x≷t 5 7 9 Stat
... saisie point 5 (97.2, 579)

1 0 0 . 1 x≷t 6 0 5 Stat
... saisie point 6 (100.1, 605)

RST ... initialisation du pointeur de programme

R/S ... calcul du coefficient m = 7.4734325186

R/S ... calcul du coefficient b = -148.5063762

9 7 SBR 1 ... pour X = 97 Y = 576,41657811

1 0 4 SBR 1 ... pour X = 104 Y = 628,73060574

5 8 0 SBR 2 ... pour Y = 580 est X = 97,479488091

6 3 0 SBR 2 ... pour Y = 630 est X = 104,16985425

Registres :

R0 ... nombre d'éléments N
R1 ... somme de y
R2 ... somme de y^2
R3 ... somme de x
R4 ... somme de x^2
R5 ... somme de x*y
R7 ... registre T
R8 ... coefficient calculé 'm'
R9 ... coefficient calculé 'b'

Programme :

- calcul de 'm'

00	33 5	RCL 5	rappel somme de x*y
01	65	-	
02	33 3	RCL 3	rappel somme de x
03	55	x	
04	33 1	RCL 1	rappel somme de y
05	45	:	
06	33 0	RCL 0	rappel nombre d'articles N
07	85	=	(somme(x*y) - somme(x)*somme(y)/N)
08	45	:	
09	43	(	
10	33 4	RCL 4	rappel somme de x^2
11	65	-	
12	33 3	RCL 3	rappel somme de x
13	23	x^2	
14	45	:	
15	33 0	RCL 0	rappel nombre d'articles N
16	85	=	
17	32 8	STO 8	mémorisation du coefficient m
18	81	R/S	arrêt du programme, affichage de m

- calcul de 'b'

19	33 1	RCL 1	rappel somme de y
20	65	-	
21	33 8	RCL 8	rappel coefficient m
22	55	x	
23	33 3	RCL 3	rappel somme de x
24	85	=	
25	45	:	
26	33 0	RCL 0	rappel nombre d'articles N
27	85	=	
28	32 9	STO 9	mémorisation du coefficient b
29	81	R/S	arrêt du programme, affichage b

- calcul de y à partir de x

30	86 1	Lbl 1	
31	43	(	
32	14	CE	
33	55	x	
34	33 8	RCL 8	rappel coefficient m
35	75	+	
36	33 9	RCL 9	rappel coefficient b
37	44	)	
38	-61	INV SBR	

- calcul de x à partir de y

39	86 2	Lbl 2	
40	43	(	
41	43	(	
42	14	CE	
43	65	-	
44	33 9	RCL 9	rappel coefficient b
45	44	)	
46	45	:	
47	33 8	RCL 8	rappel coefficient m
48	44	)	
49	-61	INV SBR	