

PROGRAMMABLE CALCULATOR

ET-58



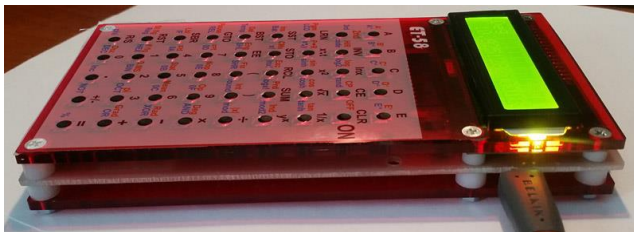
User Guide

Programmable Calculator ET-58, User Guide

Version of the Guide 1.0

October 2020

Related to calculator version 201005



website: <http://www.eternal59.com/>
mirror site: <http://www.breatharian.eu/et58/>

CONTENT

1. Keyboard Layout Map	10
2. Characteristics	12
3. Description	14
4. How to Use the Calculator	15
5. Differences from the TI-58/59	17
6. Number Format	22
7. Keyboard	23
8. Indicators on the Display	24
9. Error Indication	26
10. Number Editor	27
11. Numeric Expressions	28
12. Indirect Addressing	29
13. Programming	31
14. Buttons and Instructions	34
00 ... 09 Base digits, 0...9	34
0A ... 0F Hexadecimal digits, 0A...0F	34
10 ... 1F Letter label, A...F	35
20 Turn off the calculator, OFF	36
21 Alternative function, 2nd	37
22 Inversion of a function, INV	38
23 Natural logarithm and exponent, ln x.....	38
24 Clear error, CE	39
25 Clear display, CLR.....	40
26 Subroutine with indirect address, SBR Ind.....	40
27 Indirect internal instruction, HIR Ind	41
28 Decimal logarithm and exponent, log	41
29 Clear program and register T, CP	42
2B Enter instruction code, code	42
2C Binary logarithm and exponent, log ₂	43
2D Random number generator, rand	44
30 Tangent, tan	45
31 Programming, LRN.....	45
32 Exchange of registers X and T, x<>t.....	46

33 Square of the number, x^2	46
34 Square root of the number, \sqrt{x}	46
35 Reciprocal of the number, $1/x$	47
36 Library program selection, Pgm	47
37 Conversion of Cartesian and polar coordinates, P->R.....	48
38 Sine, sin	49
39 Cosine, cos.....	50
3A Temperature, Temp	50
3B Exchange of X and Y registers, $x \leftrightarrow y$	51
3C Hyperbolic sine, sinh.....	52
3D Hyperbolic cosine, cosh	52
3E Hyperbolic tangent, tanh	52
40 Indirect addressing, Ind	53
41 Program step forward, SST	53
42 Store number to data register, STO	54
43 Retrieving number from data register, RCL	54
44 Add and subtract number from data register, SUM.....	55
45 Power and root, y^x	55
46 Insert empty byte into program, Ins.....	56
47 Clearing data registers, CMs.....	56
48 Exchange number with data register, Exc.....	56
49 Multiply and divide data register, Prd	57
4A Battery voltage detection, Bat	57
4B Factorial, $x!$	58
4C Natural logarithm of factorial, $\ln x!$	58
4D Decimal logarithm of factorial, $\log x!$	59
4E Modulo floor, mod2	60
50 Absolute value, $ x $	61
51 Step the program back, BST	61
52 Exponent mode, EE.....	62
53 Left parenthesis, (.....	62
54 Right parenthesis,)	62
55 Division, :	63
56 Delete a byte from the program, Del	63
57 Technical mode, Eng.....	63
58 Rounding, Fix	64
59 Integer number, Int	65
5A Adjust the display contrast, LCD	65
5B Shift left, SHL	66
5C Shift right, SHR	66
5D Rounding, round	67
5E Modulo trunc, mod	68
60 Degrees, Deg	68
61 Jump, GTO	69

62 Indirect selection of a library program, Pgm Ind.....	69
63 Indirect exchange number with data register, Exc Ind	70
64 Indirect multiply and divide data register, Prd Ind	70
65 Multiplication, x	71
66 Delay, Pause	71
67 Equal, x=t.....	72
68 Žádná operace Nop.....	73
69 Speciální operace Op	73
6A Relativní skok REL.....	73
6B Nepřímá inkrementace/dekrementace registru Inc Ind.....	74
6C Nepřímá operace s registry Reg Ind.....	75
6D Nepřímá podmínka IF Ind	75
6E Bitový součin AND	76
70 Radiány Rad.....	77
71 Podprogram SBR	77
72 Nepřímé uložení čísla do registru STO Ind	78
73 Indirect retrieving number from data register, RCL Ind.....	78
74 Indirect add and subtract number from register, SUM Ind	79
75 Odečtení -	79
76 Návěští Lbl.....	79
77 Větší nebo rovno $x \geq t$	80
78 Statistika Stat.....	81
79 Průměr Mean	82
7A Podmíněný skok IF	83
7E Bitový exkluzivní součet XOR	85
80 Grady Grad.....	86
81 Reset RST	86
82 Interní instrukce HIR.....	87
83 Nepřímý skok GTO Ind.....	89
84 Nepřímá speciální operace Op Ind.....	90
85 Přičtení +	91
86 Nastavení a nulování přepínače StFlg	91
87 Test přepínače IfFlg.....	92
88 Převody minut a sekund DMS	93
89 Ludolfovo číslo pi.....	94
8A Operace s registry Reg	94
8B Hexadecimální mód HEX.....	96
8C Binární mód BIN.....	97
8D Oktalový mód OCT	97
8E Bitový součet OR	98
91 Start a stop programu R/S.....	98
92 Návrat z podprogramu RTN	99
93 Desetinná tečka	99
94 Změna znaménka +/-	99

95 Provedení výpočtu =	100
97 Programová smyčka Dsz	101
9A Zlatý řez phi.....	103
9B Dekadický mód DEC.....	103
9C Inkrementace a dekrementace registru Inc	103
9D Bitová inverze NOT.....	104
9E Procenta %.....	105
15. Speciální operace Op	107
Op 00 Vymazání tiskových registrů 1 až 4	107
Op 01..04 Nastavení tiskového registru 1..4	107
Op 09 Načtení knihovního programu	108
Op 0A Výstup registrů 1 a 2 na 1. řádek za běhu	108
Op 0B Výstup registru 1 s X na 1. řádek za běhu	108
Op 0C Výstup půl-registru 1 s X na 1. řádek za běhu	109
Op 0D Výstup registrů 3 a 4 na 2. řádek za běhu	109
Op 0E Výstup registru 3 s X na 2. řádek za běhu	110
Op 0F Výstup půl-registru 3 s X na 2. řádek za běhu	110
Op 10 Funkce sign	110
Op 11 Variace.....	111
Op 12 Koeficienty lineární regrese	111
Op 13 Korelační koeficient	112
Op 14 Lineární regrese Y z X.....	113
Op 15 Lineární regrese X z Y	113
Op 16, Op 17 Organizace paměti.....	114
Op 18 Nastavení přepínače 7 bez chyby	114
Op 19 Nastavení přepínače 7 při chybě.....	114
Op 1A Výstup registrů 1 a 2 na 1. řádek při zastavení.....	114
Op 1B Výstup registru 1 s X na 1. řádek při zastavení	114
Op 1C Výstup půl-registru 1 s X na 1. řádek při zastavení	115
Op 1D Aktivace módu displeje 'Přepínače'	115
Op 1E Aktivace módu displeje 'Registr T'	115
Op 1F Nastavení módu displeje 'Text'.....	116
Op 20 až Op 2F Inkrementace datového registru	116
Op 30 až Op 3F Dekrementace datového registru	116
Op 40 Vstup klávesy z klávesnice	116
Op 41 Test stisku tlačítka	117
Op 42 Zobrazení jednoho znaku na displeji	118
Op 43 Načtení fontu	119
Op 44 Prodleva.....	122
Op 45 Zobrazení ukazatele zleva na 1. řádku za běhu.....	122
Op 46 Zobrazení ukazatele zleva na 2. řádku za běhu.....	123
Op 47 Zobrazení textu s ukazatelem zleva při zastavení	124
Op 48 Zobrazení ukazatele zprava na 1. řádku za běhu	124

Op 49 Zobrazení ukazatele zprava na 2. řádku za běhu	125
Op 4A Zobrazení textu s ukazatelem zprava při zastavení.....	126
Op 4B Zobrazení sloupce grafu za běhu.....	127
Op 4C Zobrazení sloupce grafu s textem po zastavení	128
Op 4D Nastavení pixelu.....	129
Op 4E Vymazání pixelu	129
Op 4F Přepnutí pixelu.....	129
Op 50 Vyhledání největšího společného dělitele	130
Op 51 Načtení registru generátoru náhody	130
Op 52 Nastavení registru generátoru náhody	131
Op 53 Načtení předdefinovaného textu.....	131
Op 54 Přidání čísla k textu	132
Op 55 Inicializace zásobníku komplexních čísel a zlomků	133
Op 56 Zjištění počtu čísel v zásobníku komplexních čísel.....	134
Op 57 Vložení čísla do zásobníku komplexních čísel	134
Op 58 Načtení čísla ze zásobníku komplexních čísel.....	134
Op 59 Zrušení čísla ze zásobníku komplexních čísel	134
Op 5A Záměna dvou čísel v zásobníku komplexních čísel.....	134
Op 5B Duplikace čísla v zásobníku komplexních čísel	135
Op 5C Součet dvou komplexních čísel $X+Y$	135
Op 5D Rozdíl dvou komplexních čísel $X-Y$	135
Op 5E Násobek dvou komplexních čísel $X*Y$	135
Op 5F Podíl dvou komplexních čísel X/Y	135
Op 60 Umocnění dvou komplexních čísel X^AY	135
Op 61 Odmocnění dvou komplexních čísel $X^{(1/Y)}$	136
Op 62 Logaritmus dvou komplexních čísel $\log Y(X)$	136
Op 63 Druhá mocnina komplexního čísla X^2	136
Op 64 Druhá odmocnina komplexního čísla VX	136
Op 65 Převrácená hodnota komplexního čísla $1/X$	136
Op 66 Přirozený exponent komplexního čísla e^AX	137
Op 67 Přirozený logaritmus komplexního čísla $\ln(X)$	137
Op 68 Sinus komplexního čísla $\sin(X)$	137
Op 69 Kosinus komplexního čísla $\cos(X)$	137
Op 6A Tangens komplexního čísla $\tan(X)$	137
Op 6B Arkus sinus komplexního čísla $\asin(X)$	137
Op 6C Arkus kosinus komplexního čísla $\acos(X)$	137
Op 6D Arkus tangens komplexního čísla $\atan(X)$	138
Op 6E Konverze komplexního čísla na polární	138
Op 6F Konverze polárního čísla na komplexní	138
Op 70 Vyhledání průchodu nulou funkce A'	138
Op 71 Simpsonův integrál funkce A'	139
Op 72 Konverze úhlu z aktuální úhlové jednotky na radiány	139
Op 73 Konverze úhlu z radiánů na aktuální úhlovou jednotku	139
Op 74 Normální distribuce pravděpodobnosti $Z(x)$	140

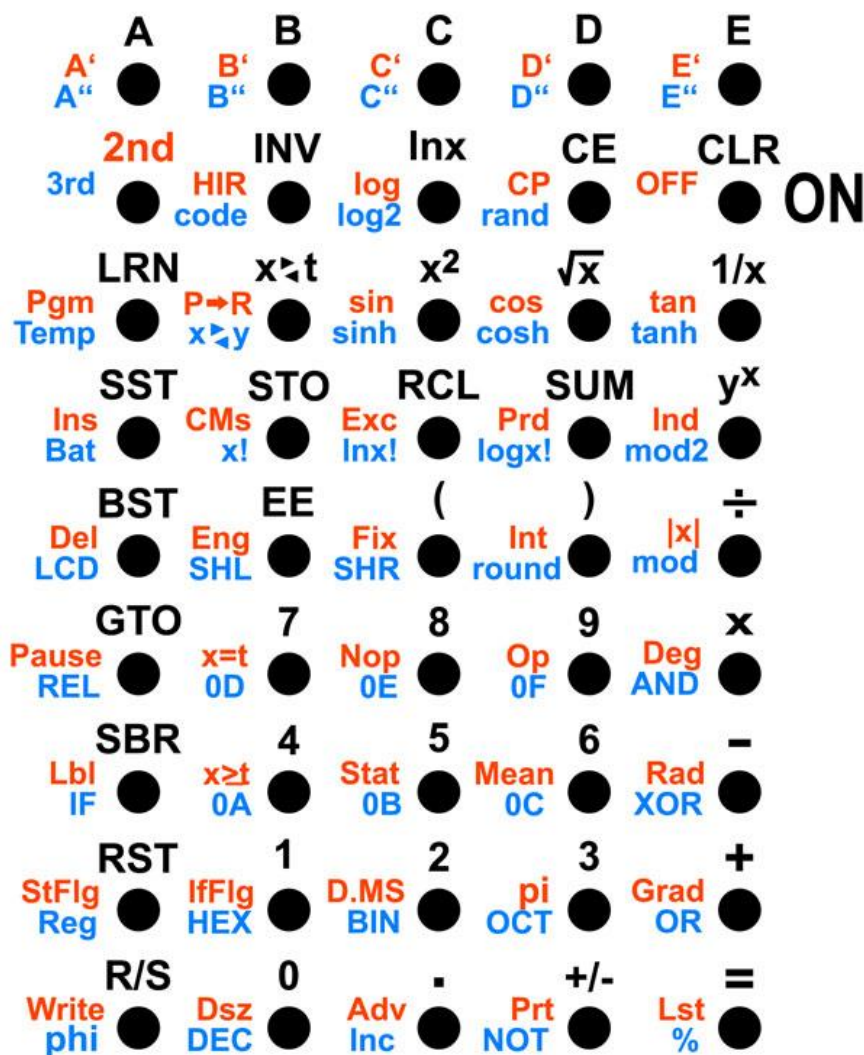
Op 75 Komplementární Gaussova distribuce $Q(x)$ CGD	140
Op 76 Kumulativní normální distribuce $P(x)$ CND	140
Op 77 Maximum	140
Op 78 Minimum	140
Op 79 Vynulování všech HIR registrů	140
Op 7A Konverze desetinného čísla na zlomek	140
Op 7B Převod zlomku na desetinné číslo	141
Op 7C Součet zlomků $X+Y$	141
Op 7D Rozdíl zlomků $X-Y$	141
Op 7E Násobek dvou zlomků $X*Y$	141
Op 7F Podíl dvou zlomků X/Y	141
Op 80 Krátká prodleva 10 msec	142
Op 81 Krátká prodleva 100 msec	142
Op 82 Odstranění skrytých číslic	142
Op 83 Zahájení měření času	142
Op 84 Zjištění uplynulého času	142
Op 85 Zjištění počtu datových registrů	143
Op 86 Zjištění stavu uživatelských přepínačů	143
Op 87 Výpočet kontrolního součtu ROM paměti	143
Op 88 Nastavení prodlevy pro vypnutí kalkulátoru	143
Op 89 Zjištění prodlevy pro vypnutí kalkulátoru	143
Op 8A Zobrazení verze firmware kalkulátoru	144
Op 8B Reset kalkulátoru	144
16. Character Table	145
17. Knihovny programy	146
ML-01 Diagnostika	147
ML-02 Determinant matice	149
ML-03 Sčítání a násobení matic	153
ML-04 Komplexní aritmetika	157
ML-05 Komplexní funkce	160
ML-06 Komplexní trigonometrické funkce	163
ML-07 Vyčíslení polynomu	166
ML-08 Průchod funkce nulou	167
ML-09 Simpsonův integrál funkce	170
ML-10 Simpsonův diskrétní integrál	172
ML-11 Řešení trojúhelníků zadaných stranami	173
ML-12 Řešení trojúhelníků zadaných úhly	176
ML-13 Kruhový oblouk	179
ML-14 Normální rozdělení	181
ML-15 Generátor náhodných čísel	183
ML-16 Variace, permutace, kombinace, faktoriál	184
ML-17 Klouzavý průměr	189
ML-18 Složený úrok	190

ML-19 Splátky.....	193
ML-20 Den v týdnu, dny mezi daty	197
ML-21 Hra HI-LO	199
ML-22 Běžný a spořicí účet.....	200
ML-23 DMS operace	203
ML-24 Konverze jednotek 1	205
ML-25 Konverze jednotek 2	206
ML-26 Aritmetika zlomků	207
ML-27 Astabilní generátor s obvodem 555	211
ML-28 (EE-07) Konverze poměrů	214
ML-29 (EE-11) Reaktance LC	215
ML-30 (EE-12) Konverze sériové/paralelní impedance	217
ML-31 (EE-13) Aktivní filtr	219
ML-32 (EE-14) Pasivní dolní propust	223
ML-33 (EE-15) Konvoluce signálu.....	226
ML-34 (EE-17) Diskrétní Fourierova transformace	229
ML-35 Ohmův zákon	232
ML-36 Sérové a paralelní řazení	234
ML-37 (AV-23) Konverze časové zóny.....	235
ML-38 (MU-06) Shellovo třídění	238
ML-39 (MU-09) Rozklad na prvočinitele.....	240
ML-40 (MU-21) Aritmetika s proměnnými	241
ML-41 (MU-14) Interpolace	242
ML-42 (MU-16) Vyhledání minima a maxima.....	244
ML-43 Kontrolní součet	246
ML-44 (LE-09) Hra Codebreaker.....	250
ML-45 (LE-12) Hra Acey-Deucy	251
ML-46 (LE-13) Hra Craps.....	253
ML-47 (LE-14) Hra Přistání na Marsu	255
ML-48 Hra Nim	257
ML-49 Měření reakčního času, stopky	258
ML-50 (LE-20) Hra Námořní bitva	259

1. Keyboard Layout Map

For each button, basic meaning is given on the 1st row, first alternative meaning on the 2nd row (after pressing the **2nd** button) and second alternative meaning on the 3rd row (after pressing the **2nd** **2nd** button twice, ie **3rd**).

[11] A	[12] B	[13] C	[14] D	[15] E
[16] A'	[17] B'	[18] C'	[19] D'	[10] E'
[1A] A''	[1B] B''	[1C] C''	[1D] D''	[1E] E''
[21] 2nd	[22] INV	[23] ln x	[24] CE	[25] CLR
[]	[82] HIR	[28] log x	[29] CP	[20] OFF
[]	[2B] code	[2C] log2	[2D] rand	[2E]
[31] LRN	[32] x<>t	[33] x^2	[34] Vx	[35] 1/x
[36] Pgm	[37] P->R	[38] sin	[39] cos	[30] tan
[3A] Temp	[3B] x<>y	[3C] sinh	[3D] cosh	[3E] tanh
[41] SST	[42] STO	[43] RCL	[44] SUM	[45] y^x
[46] Ins	[47] CMs	[48] Exc	[49] Prd	[40] Ind
[4A] Bat	[4B] x!	[4C] ln x!	[4D] log x!	[4E] mod2
[51] BST	[52] EE	[53] ([54])	[55] :
[56] Del	[57] Eng	[58] Fix	[59] Int	[50] lxl
[5A] LCD	[5B] SHL	[5C] SHR	[5D] round	[5E] mod
[61] GTO	[07] 7	[08] 8	[09] 9	[65] x
[66] Pause	[67] x=t	[68] Nop	[69] Op	[60] Deg
[6A] REL	[0D] 0D	[0E] 0E	[0F] 0F	[6E] AND
[71] SBR	[04] 4	[05] 5	[06] 6	[75] -
[76] Lbl	[77] x>=t	[78] Stat	[79] Mean	[70] Rad
[7A] IF	[0A] 0A	[0B] 0B	[0C] 0C	[7E] XOR
[81] RST	[01] 1	[02] 2	[03] 3	[85] +
[86] StFlg	[87] IfFlg	[88] DMS	[89] pi	[80] Grad
[8A] Reg	[8B] HEX	[8C] BIN	[8D] OCT	[8E] OR
[91] R/S	[00] 0	[93] .	[94] +/-	[95] =
[96] Write	[97] Dsz	[98] Adv	[99] Prt	[90] Lst

[9A] [phi](#)[9B] [DEC](#)[9C] [Inc](#)[9D] [NOT](#)[9E] [%](#)

2. Characteristics

- ATmega328P processor (4 MHz, 32 KB ROM, 2 KB RAM)
- Supply voltage 2.5 V (from battery) up to 5.5 V (from USB connector).
- Precision of calculations 19 digits
- Display number up to 14 significant digits
- Exponent 4 digits, range ± 9863
- 1000 program steps of user program in EEPROM (saves program even without battery)
- 110 data registers
- 16 control HIR registers
- Two-line LCD display (2 x 16 alphanumeric characters)
- 45 key buttons
- Calculator code completely written in AVR assembler
- Built-in library of 50 programs with total length of almost 10,000 program steps
- Exponential and logarithmic functions
- Trigonometric functions
- Hyperbolic functions
- Factorial of decimal and large numbers
- Random number generator
- Indexed access to variables
- Indirect parameters of functions
- Scientific and technical display mode with exponent
- Display mode HEX, OCT a BIN, including decimals
- Bitwise operations AND, OR, XOR, NOT, shifts
- Pseudographic display of graphs and indicators
- Programmatic display of text
- Dynamic input from keyboard while program is running
- Statistical functions and linear regression
- Absolute addressing, labels, relative jumps
- Calculations with matrices
- Complex numbers
- Fractions
- Enumeration of polynomials
- Numeric search of function roots
- Numerical calculation of integrals
- Interpolation and approximation

- Triangle calculations
- Unit conversions
- Circular sectors
- Combinations, permutations
- Floating mean
- Interests and repayments
- Time transfers, time zones
- Interval between dates, day of the week
- Games (Hi-Lo, Codebreaker, Acey-Deucey, Lander and more)
- Astable generator with 555
- Reactance of capacitors and inductors
- Serial and parallel connection of components
- Calculations of active and passive filters
- Signal convolution
- Discrete Fourier transform
- Ohm's law
- Sorting numbers and median
- Search for prime factors
- Search for minimum and maximum of function
- Measurement of reaction time

3. Description

The ET-58 calculator is intended primarily for handyman and those who interested in retro computer technology. Conceptually, it is based on the popular TI-58/59 calculator, developed in 1978 by the Texas Instruments company. It expands its features, while trying to preserve the functionality of the original calculator as much as possible. The ET-58 calculator is offered in the form of a kit. It focuses on the cheapest possible design and easy assembly of the kit, so it uses a maximally simple construction without button keycaps and with a minimum of SMD components.

4. How to Use the Calculator

The ET-58 calculator is equipped with a 2-row alphanumeric LCD display, 45 microswitches, a processor and a battery.

The calculator is switched on by pressing the **CLR** button (alternative function **OFF**). It is switched off either by pressing **OFF** (ie by pressing the **2nd CLR** buttons) or automatically after selected time of calculator inactivity (default set to 30 seconds - it can be changed by **Op 88**). If the battery is not removed for more than a few minutes (and external power supply disconnected), the contents of the memory registers, displayed number and initiated operations (ie contents of the RAM memory) will be retained unchanged. For calculator without a battery inserted, do not press the **CLR** button to keep the RAM contents after power-up, otherwise the calculator will reset after power-up and forget the RAM contents. The loss of memory content does not affect the program in ROM (library) and EEPROM (user program). They remain unchanged even without a battery.

If battery will be removed while the calculator is not turned off, the RAM contents may be lost immediately. In this way, you can reset the calculator to its default state if it exhibits incorrect behavior or if you want to reset the calculator into default settings. It is also possible to return calculator to default settings by the **CE** subroutine of library program 1 (to do it, enter **Pgm 0 1 SBR CE RST**).

The contents of the user program and the uploaded user library will not be lost even after a long battery removal.

Caution - Do not remove the battery or disconnect power during program writing operations or while loading library into memory. This could damage the contents of the memory.

The contrast of LCD display depends on the supply voltage. Use the **LCD** button to adjust the display contrast (you can invoke it up by pressing the **2nd 2nd BST** buttons). After pressing the button, enter the number **0** to **9** into the calculator, which represents the desired contrast of the LCD display. The number 0 means minimum contrast (characters on the display are gray or even barely visible), the number 9 means maximum contrast (there are dark rectangles below the characters on the display). By changing the contrast of the display, you can get outside the viewable

range, when you will no longer recognize the text on the display. Even in this case, try changing the contrast with the **LCD** button. If no characters are visible on the display (the calculator looks switched off), try setting a higher display contrast from the keyboard using a higher number. Alternatively, press **CLR** first, the calculator may be switched off. Conversely, if there are black rectangles on the display, set a lower value of the display contrast. If you do not know in what state the calculator was in lasted (it may be now in programming mode), remove battery, press **CLR** to reset the calculator. Then insert the battery, try turning on the calculator by the **CLR** button, and set the display contrast. If it fails, try a new battery or external power supply, maybe the battery is low.

After resetting the calculator (the first time it is turned on after inserting the battery), the calculator will show its name for 2 seconds, along with a 6-digit code representing the calculator firmware version date. E.g. "ET-58 201005" means firmware date (build) 10/05/2020. To view the firmware version again, remove and re-insert the battery into the calculator without turning calculator off. This will reset the calculator. If you remove battery from the calculator while it is turned off, the calculator can keep the data for an hour without performing a reset after power-up. In that case press **CLR** button and the reset will occur even when it is turned off. You can also use **Op 8A** or **Pgm 01 A'** to display the firmware version.

Each time the calculator is turned on, it checks the integrity of the internal ROM using a 16-bit checksum (CRC-XModem). If an internal memory error occurs, the calculator displays a "CRC Error" warning for 2 seconds. The calculator can still be used, but its processor is obviously defective and may show unpredictable activity.

5. Differences from the TI-58/59

Although software of the ET-58 calculator strives for maximum compatibility with the original TI-58/59 serie, full compatibility cannot be ensured and then some programs may need to be modified during import.

Higher accuracy

Original TI-58/59 calculators worked with numbers in BCD code (1 byte = 2 decimal digits) and with internal precision of 13 digits. Size of the number was 10 bytes. Although ET-58 also uses 10-byte number, it uses binary code, which allows it to achieve higher accuracy with 19-digit mantissa and faster calculations. Higher accuracy is usually not a problem, but using a binary code instead of a BCD code can result in a loss of accuracy for numbers with a finite number of digits. The calculator tries to correct deflections, but sometimes they can appear themselves and it may be necessary to take them into account.

A typical example is integers used as a counter in loops. After thousands of cycles of adding/subtracting the value 1, a small deviation from the integer number may occur, which may not appear on the display, but it may occur while comparing the number to a match. The calculator treats this deviation so that the loop function (DSZ, DJNZ, DJZ) rounds the number to an integer after each decrement.

Although the calculator takes into account a certain tolerance for deviation, it may be necessary to take into account an increasing deviation from the assumed exact value for repeated calculations and to use rounding to an integer or compare in the interval with tolerance before comparing to a match.

Furthermore, the different accuracy proves in the random number generator. The original TI-58/59 random number generator performs addition and multiplication operations and moves hidden digits into visible positions. This is reflected in the fact that the calculation of the random number of the ET-58 deviates completely from the random generator of the original TI-58/59 calculator after a few steps. Usually this fact should not matter. In addition, in the ET-58 programs, the original random generator is no longer used, the calculator has an internal random generator with a significantly better randomness distribution.

Hexadecimal code

The program code is stored in the BCD code for the original TI-58/59 calculator. Each byte is divided into halves containing 2 digits with a value of 0 to 9. The two digits thus represent the decimal value 0 to 99. In contrast, the code of the ET-58 calculator is stored in memory in hexadecimal code. It is again expressed using 2 digits, but in the range 0 to 15. Digits with a value of 10 to 15 are expressed by the letters A to F. Hexadecimal notation supports both the original meaning, ie writing the value 00 to 99 (ie 100 values) using 2 digits 0 to 9, as well as a hexadecimal meaning with a value of 00 to FF (i.e. 256 values). The interpretation of the code depends on the meaning of the byte.

The original codes of the buttons are retained. E.g. the **CLR** button has the code 25 here too, as with the original TI-58/59, regardless of whether it is a binary or hexadecimal notation. Hexadecimal characters extend the code set by codes with characters A to F.

The absolute address and register number are expressed by 2 decimal digits 0 to 9, as with the original TI-58/59, and has a value of 0 to 99. For register numbers of the calculator, the number A can also be used as the first character and thereby increasing the number of addressable registers from 100 to 110.



In some special cases, the program code has the meaning of 2 hexadecimal digits. This is, for example, the HIR instruction code. The first digit 0 to F represents the operation code, the second digit 0 to F represents the HIR register number 0 to 15.

Rounding the loops

The DSZ loop of the original calculator decrements the value of the register by 1, and if it does not reach zero, it jumps back. As mentioned above, due to the binary expression of the number, the register value of the ET-58 may deviate slightly from the integer number, and this would lead to inaccuracy of the loop. The calculator treats this by rounding the result to an integer after each decrement. This can occur if the original TI-58/59 program envisages the use of decimal numbers.

The same applies to the DJNZ and DJZ instructions (group of HIR instructions), which, however, the original calculator does not have.

Repeat calculation

The ET-58 calculator repeats the last entered arithmetic operation after pressing the  key. Some programs of the original TI-58/59 do not count on repetition of operations, they use the  key several times and this may result in an incorrect calculation.




Higher speed






The ET-58 calculator uses a more modern and powerful processor, with a higher clock frequency. As a result, the calculations are noticeably faster than with the original calculator. This feature is usually not a problem. It can occur in programs that suppose speed of original calculator, such as a perception test program. But there are a negligible number of such programs.

HIR registers

The original TI-58/59 calculator has special internal HIR instructions, used by the internal code and not published in the user manual. HIR instructions originally used arithmetic operation registers 0 through 9. For the ET-58, HIR instructions are validated as valid instructions and are even used by libraries as the main working registers. Unlike the original calculator, they are not shared with the arithmetic unit. These are full-fledged registers, not rewritten by any other operations.

Fix rounding

For the original TI-58/59 calculator, the  key is used to set the number of displayed decimal places to 0 to 8. A value of 9 turns off rounding, the number is displayed with maximum number of decimal places.   has the same meaning.

The ET-58 calculator allows to display number with up to 13 decimal places. Therefore, the parameter of the  instruction is extended to the values 0 to 9 and A to D, representing rounding to 0 to 13 decimal places. The   code is used to turn off the rounding, as is the   command.

For older programs,   is often used to turn off the rounding. In such

cases, the code must be changed to **INV** **Fix** or **Fix** **OF**.

Error indication

With the original TI-58/59 calculator, the display blinking is sometimes used to indicate program error. Code **0** **1/X** is used for this purpose. Such code will work without change for the ET-58.

The second case, sometimes used, is the code **+** **=**. For the original calculator, this sequence ensures that the data shown on the display blinks. For ET-58, this sequence does not activate an error indication. The indication can be replaced either by the sequence **X/T** **0** **1/X** **X/T** or by using **StFlg** **OF**, which turns on switch 15, which is also the state of the error indication.

INV transfer

The original TI-58/59 calculator sometimes uses the **INV** (inverse function) prefix setting over several instructions. The calculator assumes that instructions that do not use this prefix do not change its state. With the ET-58 calculator, many functions are extended by an inverse or alternative function, so most commands process the **INV** code and do not save it.

The most common use of **INV** transfer is for labels. The first label is followed by the **INV** command itself, followed by the second label and the first command. In such case, running the program from the first label invokes the inverse function of the command, running the program from the second label invokes the non-inverted function. In this case, the state of the **INV** button is maintained during the **LBL** label at the ET-58 calculator, too.

2nd INV sequence

For the original TI-58/59, the order of pressing the **2nd** and **INV** buttons did not matter. In both cases, an inverse alternative function was performed. When writing to the program, in one case the code **INV** and the alternative code of the button were written, in the other case the code **2nd** **INV** (code 26) and the normal code of the button were written - the change to the alternative code was made until program execution.

This cannot be used with the ET-58 calculator and it is necessary to maintain the order of pressing the **INV** **2nd** **...** buttons. The **INV** button first sets the prefix of the inverse function and the **2nd** **...** button writes an alternative button code to the program. When pressed in the opposite direction, **2nd** **INV**, an alternative function of the **INV** button is performed, ie the **HIR** function.

Character table

The ET-58 uses a different character table for printing and displaying than the TI-58/59. Characters are entered as 2 digits of a decimal number in the range 00 to 99 and (except for small deviations) they correspond to ASCII characters decreased by 32. When importing programs, it is necessary to change the contents of registers **Op** **01** to **Op** **04**. For more details see chapter [Character Table](#).

HIR 20 function

The **HIR** **20** function was used to branch the internal microcode program on the original TI-58/59. It allowed, according to a preset internal register, to perform either a relative jump or termination of a function. At ET-58 it loses its significance and it is therefore used for other purposes (rounding of the HIR register).

Memory organization

In the original TI-58/59, RAM was divided between program memory and data register memory. The partition could be changed by operation **Op** **17**. The partition cannot be set with the ET-58 and always corresponds to a maximum of 1000 program steps (in the EEPROM memory) and 100 or more data registers (in the RAM memory).

Negative exponent overflow

For the original TI-58/59, an exponent overflow to large negative values was indicated as an underflow error (flashing 1-99). For ET-58, the overflow of the large negative exponent is considered the number 0 and the error is not indicated.

6. Number Format

The user does not usually encounter the internal number format. It can be encountered if the mantissa is displayed with the **INV DEC** command.

The numbers in calculator are stored in 10 bytes of memory. The first two bytes contain exponent, with higher byte at lower address. The exponent is an unsigned integer, with a bias of 0x8000 (32768) and with a range of valid values from 0x0001 to 0xFFFFE. Exponent with a value of 0x0000 is a special case and represents zero number (the content of the mantissa does not matter). The second special case is value 0xFFFFF, representing infinity (overflow). After converting to a decimal number form, the exponent has a value in the range -9863 to +9863.

8 bytes are reserved for the mantissa. Mantissa is a binary number, with the most significant byte at the lower memory address (ie offset 2). The highest bit of the mantissa (ie bit 7 of the first byte at offset 2) always has the value 1 for a non-zero number and is therefore not expressed in the number, it is hidden. Its position is used to indicate the sign of the number (1 means negative number).

Accuracy of the mantissa is 19.27 decimal digits. The display shows a maximum of 14 digits, ie 5 digits are hidden and serve to maintain the accuracy of calculator calculations. The popular trigonometric test can be used to test the accuracy of the calculator:

9 sin cos tan INV tan INV cos INV sin

If calculated correctly, the result should again be number 9. The calculation quickly loses accuracy and a deviation usually occurs in calculators. For the ET-58, the deviation remains within the hidden digits and the displayed result will again be the number 9.

More on calculator accuracy: <http://www.datamath.org/Forensics.htm>

7. Keyboard

The calculator can be operated either in direct mode, where the key codes are executed immediately, or in programming mode, where the key codes are only written to the program but not executed.

The calculator is controlled by a set of 45 buttons, arranged in 9 rows and 5 columns. Rows are numbered from top to bottom, in order 1 to 9. Columns are numbered from left to right, with numbers 1 to 5.

After pressing the **2nd** button, the alternative function of the button is used, indicated by the column number 6 to 10 (number 10 is replaced by the digit 0 in the code). After second pressing of the **2nd** button (ie meaning **3rd**), the second alternative function is used. It is indicated by column A to E.

When writing a program to the memory (using the **LRN** key), the key code is written to the program as a pair of digits, where the first digit represents the row of the key (typically 1 to 9) and the second digit the column of the key (typically 1 to 5 for the basic function or 6 to E for alternative function).

Codes of numeric keys **0** to **9** are not stored in the program using the key coordinate, but as a decimal value 00 to 09.

In addition to the listed key codes, there may be secondary key function codes and special commands in the program that use codes that are not directly accessible from the keyboard. E.g. the **INV** **SBR** sequence is stored in the program as **RTN** command with code 92.

Note: In the text of the manual, the names of the buttons are given without any 2nd prefixes, which may be necessary to call function of the button. E.g. the **rand** key code (random number) is called up by pressing the **2nd** **CE** keys.

8. Indicators on the Display

LCD display contains 2 rows of 16 alphanumeric characters. First row is typically used to display indicators, second row to display edited number and result of operation.

Meaning of first row of the display can be switched by instructions:

Op 1D ... indicators (default mode)
Op 1E ... register T
Op 1F ... text (mode can be turned off using **CLR**)



Indicators at first row of the display:

Deg/Rad/Grd - indication of unit of angles in degrees, radians or grads. $360^\circ = 2 * \pi$ radians = 400 grads. The switch can be changed with the **Deg**, **Rad** or **Grad** buttons.

Hex/Bin/Oct - indicates selected numeral system: decimal (not indicated), hexadecimal (Hex), binary (Bin) and octal (Oct). Numeral system can be selected with **DEC**, **HEX**, **BIN** and **OCT** buttons.

F0 to FD - indicates selected rounding of numbers from 0 to 13 decimal places. It is set with the buttons **Fix 0** to **Fix 0D**. The **INV Fix** sequence (**Fix 0F** has the same meaning) turns off the rounding of displayed result. In this case, rounding is not indicated on the display (it is replaced by spaces).

EE/Eng - indicates how the exponent is displayed. After pressing **EE**, the number is displayed in scientific form, in mantissa and exponent format. The mode can be canceled by pressing **CLR** or **INV EE**. After pressing **2nd Eng**, the number is displayed in technical (engineering) form, where the exponent is a multiple of 3. The Eng mode is switched off by pressing

INV Eng. If the exponent modes are off, nothing is indicated on the display.

2n/3d - indicates the first or second press of **2nd**, the alternative function key. If some key is pressed after pressing the **2nd** key once, an alternative function (visible on the keyboard in the second row) is performed instead of its basic function. After pressing the **2nd** key a second time, 3d appears on the display. In this case, the second alternative function of the selected key is performed (visible on the third row of the keyboard). If the **2nd** key is not pressed, or if it is pressed 3 times, the alternative function is not active, the basic function of the button is performed. This status is not indicated on the display (spaces appear at the position).

In - indication of pressing the **INV** button, activating the inverse function. If the **INV** (inverse alternative function) key is to be pressed together with the **2nd** key, the **INV** key must be pressed before pressing the **2nd** key.

Operation - last position of 1st row is intended to indicate the selected arithmetic operation: + addition, - subtraction, * multiplication, : division, & bitwise AND, | bitwise OR, ~ bitwise XOR, \ modulo truncate (mod), / modulo floor (mod2), % percentage, < shift left, > shift right, ^ power, V root, (open parenthesis.

9. Error Indication

If an error occurs while the calculator is running, the error will be indicated by a flashing the display. At the same time, an **E** or **F** character appears at the beginning of the second row, indicating the type of error.

Error type **E** (Error) is a soft error. It is created, for example, by dividing a number by zero. The program can continue to run and the error will not be indicated until the program has finished running. The indication can be canceled with the **CE** or **CLR** button.

The behavior of the program in the event of an error can be influenced by switch 8. If switch 8 is set (instruction **StFlg 8**), the program will stop after a soft error. If switch 8 is not set (default state), the program makes only the most necessary correction and continues to run. In this case, the error will be indicated only after the end of the program.

The status of the error indication can be detected programmatically using the commands **Op 18** and **Op 19**, together with switch 7. By calling **Op 18**, switch 7 is set if the error is not indicated. Otherwise, the state of the switch does not change. Calling **Op 19** sets switch 7 if an error is indicated. Otherwise, the state of the switch does not change.

The third way of working with the error indication is switch 15. It is directly connected to the error indication and can be both detected (with the **IfFlg 0F** instruction) and set (**StFlg 0F**) or reset (**INV StFlg 0F**).

Note: After resetting the error indication by clearing the switch 15, the character **E** may remain lit on the display. This is not a fault, the character disappears after the first change of the display content.

You can also use method used in the TI-58/59 calculator programs to set the error indication. The **CLR 1/X** key sequence starts flashing the display with 9.9999+9999 number.

Type **F** (Fatal) error is a hard error that does not allow the program to continue running and leads to its immediate stop. It is created by overflowing the program pointer behind the end of memory, overflowing the depth of the program stack or the stack of operations, using a non-existing label, or using an invalid register number.

10. Number Editor

The number entered, as well as the results of the calculations, appear on the 2nd row of the display. The mantissa is displayed with an accuracy of max. 14 digits.

One position for a sign is reserved in front of the mantissa. The '-' character is displayed here for negative numbers, leaving a space for positive numbers.

An exponent is displayed behind the mantissa (if exponent mode is active). The exponent is separated from the mantissa by a + or - sign. The exponent is displayed with 1 to 4 digits.

The mantissa can include a decimal point. In exponent mode with scientific notation (mantissa and exponent), the decimal point is always displayed after the first digit. If the exponent mode is not active, a decimal point is displayed after the unit digit. If there are no digits after the decimal point, decimal point will not appear.

In technical (engineering) mode with decimal or octal mode, the exponent is multiple of 3. One to three digits are displayed before the decimal point. If hexadecimal or binary display mode is active, the exponent is a multiple of 4. One to four digits are displayed before the decimal point.

If a non-decimal numeral system is active, the mantissa digits are displayed in the selected numeral system, but the exponent is always displayed as a decimal number.

The **CE** key deletes the last character of the mantissa or exponent (depending on where the digits are being written).

The **EE** key starts entering the exponent. You can return to entering the mantissa by pressing the dot button **.** or **INV EE**. The **EE** button is also used to start editing displayed result of operation. This can be used to remove hidden digits of a number. Other alternatives are **INV |** and **Op 82**.


11. Numeric Expressions



During calculations, the calculator maintains the priority of operations in 3 levels:

1. + addition, - subtraction, & bitwise AND, | bitwise OR, ~ bitwise XOR
2. * multiplication, : division, \ modulo trunc (mod), / modulo floor (mod2), % percentage, < shift left (multiplication 2x), > shift right (division / 2)
3. ^ power, √ root

In the calculations, the level 3) power and root is evaluated first, then 2) multiplication and division, and finally 1) addition and subtraction.

You can use parentheses in the expression arbitrarily, up to the 15th level.

The  key is used to swap the first and second operands of the operation.

After performing the calculation, the lowest level of calculation can be repeated by pressing the  button again. Entering a number and pressing  repeats the operation. The entered number is used as the first operand of the operation, the second operand remains the original.

Example:

    [5]

  [6]

  [12]

12. Indirect Addressing

In addition to direct entering the parameters of instructions (their numerical value), the parameters can also be entered indirectly, using data registers. In such case, the register number from which the calculator is to take the parameter is specified as a parameter. Use the **Ind** button to change the pointer to an indirect address. In some cases, another instruction code (intended for indirect addressing) is stored to the program, in other cases, the **Ind** code is stored as an indicator of indirect addressing.

For jump instructions, the register contains the absolute jump address as a decimal number in range 0 to 999. In addition to the absolute jump address, a label symbol can also be stored to the register. The label is stored in the register as a label code, expressed by a decimal number and multiplied by 256 (label code is in higher byte of the number). E.g. the **1/x** button has a hexadecimal program code of 35. This corresponds to decimal number 53 ($3 \cdot 16 + 5 = 53$). Multiplying by 256 results in a value of 13568 (or 3500 in HEX code).

Simply placing the **Ind** command in the program executes an instruction, the code of which is stored in the data register, the code of which is given after the **Ind** instruction. The **Ind** instruction itself is not normally intended for operation with its own parameter, and therefore it is necessary to either use a one-digit register code (digits 0 to 9) or do a step back and correct the register number. The instruction code must be stored in decimal form in the register. If the instruction requires any parameters, they are read from the part of the program following the **Ind** code.

Example, indirect addressing of the register:

5 STO 01 stores value 5 to the register R01 (by direct addressing)

8 STO 05 ... stores value 8 to the register R05 (by direct addressing)

RCL Ind 01 [8] ... reads value 5 from register R01 and uses it as address of register R05, from which it reads the resulting value 8

Instead of code 43 01 (instruction **RCL 01**), the code 73 01 (instruction **RCL Ind 01**) is stored to the program.

Example, indirect absolute address:

1 2 3 STO 0 1 ... stores value 123 to the register R01

GTO Ind 0 1 ... reads value 123 from the register R01 and uses it as a jump to the new address

LRN [123 FF ...empty] ... check the current program indicator

Example, indirect label:

RST LRN ... activation of programming mode

Lbl 1/x ... write label **1/x**

LRN ... switch off programming mode

RST ... reset pointer back to 000 (to check jump execution)

1 3 5 6 8 STO 0 1 ... store value of the 1/X label to the register R01
(code of 1/x = 35 hex = 53 decimal, * 256 = 13568)

GTO Ind 0 1 ... reads the label from the register R01 and jumps to **1/x**

LRN (002 FF ...empty) ... check destination address (after label **1/x**)

Example, indirect instruction:

RST LRN ... activation of programming mode

Ind 1 2 ... indirect instruction from register R01 with parameter '2'

LRN ... switch off programming mode

4 5 STO 0 2 ... stores test value 45 into register R02

6 7 STO 0 1 ... stores decimal value of **RCL** instruction into register R01

RST ... resets the program pointer to 000

SST [45] ... executes instruction at address 000. Reads the **RCL** instruction code (43 hex = 67 dec) from register R01, activates it, the **RCL** instruction reads the parameter following the **Ind** instruction, ie the value '2'. The **RCL 02** instruction is executed, which displays the contents of the R02 register (in our case, the value 45).

LRN [003 FF ...empty] ... check destination address after codes **Ind 01 02**.

13. Programming

Writing a sequence of buttons into the program memory is called a program. With a program, the calculator becomes a powerful tool. There is a user program, writable into the EEPROM memory (ie memory, the content of which is saved in the processor even after removing the battery), and a program library, which is stored in ROM memory. The program library cannot be overwritten by the program editor.

The programming mode is activated by the **LRN** button. The program content is displayed on two rows of the display. On the bottom row from the left, there are 3 digits, representing the address of the current program step in memory. The address is in the range of 000 to 999 (ie 1000 program steps).



The two-digit HEX code of the program byte at the given address is displayed after the address. The byte code is followed by a name of the button or function that corresponds to that code. The calculator cannot distinguish in the program editor whether it is the beginning of an instruction or a parameter, and therefore displays the appropriate text of the button name for all codes. It is up to the user to distinguish according to the code of the previous instruction, whether it is an instruction code or a parameter.

The top row shows the contents of the adjacent 4 bytes of the program. This will make it easier to navigate the program. The current program byte (which is selected in the bottom row) is displayed in the middle of the top row and is additionally framed by square brackets.

The programming mode can be activated even if some program from the library is active (eg by selecting **Pgm 02**). In this case, the display does not show the user program (as with the original TI-58/59), but the contents of the library program. The program can only be scrolled through, it cannot be modified in any way. The library program is distinguished from the main

program by the fact that an asterisk * is displayed between the address and the program byte code (as an indication of the "Read-Only" mode).

Keys during programming

SST (Single Step) - Increases program pointer by 1 ("next step"). The **SST** button can also be used during normal (execution) mode. After pressing it, the code of the instruction to which the program pointer is set is executed.

BST (Back Step) - Decreases the program pointer by 1 ("reverse step").

Ins (Insert) - Inserts an empty byte at the current program position and moves away the next part of the program. An empty byte has the value FF and is marked with the label "... empty" in the editor. This byte has a similar function at runtime as the **Nop** instruction (ie nothing is done), but it has a special meaning during program editing. The **Ins** and **Del** buttons move rest of program memory behind the current program pointer. However, if an FF code is encountered, the operation is aborted. The FF code thus acts as a kind of flexible gap separating the individual parts of the program. It ensures that the following sections of the program, separated by the space FF, stay in place. This is suitable, for example, in cases of absolute addressing. In addition, it speeds up **Ins** and **Del** operations, which can take several seconds at full memory. The separator space is applied until it disappears completely (by inserting more bytes). The sections will merge and continue to move together.

Note that some instructions allow you to enter a parameter in the HEX code and insert a valid byte with a value of 0FF into the code. If possible, avoid such a byte value, as it would be interpreted as blank space and damaged during program moving in memory.

Del (Delete) - Deletes byte at current program position and pulls up following part of the program. During the operation, empty FF bytes are applied, as described in the **Ins** instruction.

It should be noted that the **Ins** and **Del** instructions do not correct the absolute jump addresses in the program. For this reason, it is better to use either relative jumps or, even better, labels. With the original TI-58/59 calculator, absolute addresses were preferred due to their higher speed, as finding labels in the program can take a long time (even a few seconds).

Opposite to this, the ET-58 searches for program labels very quickly and thus it can be used as a full-fledged replacement of absolute addresses.

If it is difficult for someone to figure out which buttons he can still use as labels in a program, he can use numeric codes of the buttons for this purpose. Codes 0A0 to 0FE are suitable for this purpose, which are not assigned to any buttons and can be used well as program labels. You can use the **code** key to enter from the keyboard, followed by the two digits of the label code.

LRN (Learn) - Exits edit mode and returns the calculator to execute mode.

GTO (Go To) - The **GTO** button cannot be used directly as a control key in the programming mode, because its code is inserted into the program when pressed. However, it can be used in execution mode by temporarily switching off the programming mode by pressing **LRN**, pressing **GTO** and entering a 3-digit numeric address or button label, and pressing **LRN** to return to the programming mode. The program pointer will be moved to the specified address.

RST (Reset) - Like the **GTO**, it cannot be used directly in programming mode, but it can be used in execution mode to rewind program pointer to address 000. If some library program is active, it is deactivated and switched back to the main user program.

R/S (Run/Stop) - Program start or program stop (used in execution mode).

Example:

Pgm **02** ... selects library program 2

GTO **1** **2** **3** ... moves program pointer to address 123

LRN [123*76 Lbl]... switches to programming mode - check address

LRN ... turns off programming mode

RST ... resets program pointer and switches to the main program



LRN [000 FF ...empty] ... switches to programming mode to check address

14. Buttons and Instructions

For each button, the program HEX code, the name of the button and the sequence of button presses to call it are given.

00 ... 09 Base digits, 0...9





Symbol  ... 




Call up with buttons  ... 

Base digits are used to enter digits in the range 0 to 9, typically a decimal number. They are used to enter the mantissa of the number, to enter exponent, number of memory register, absolute jump address, and more.

Separate digits are stored in the program with codes 0 to 9. If they are part of a compound number, such as register number or absolute jump address, they are stored in the program compounded in BCD code (ie 2 digits per byte).

Example:



The     key sequence will be stored into the program with codes 61 01 23.

The    key sequence will be stored with codes 42 12.

0A ... 0F Hexadecimal digits, 0A...0F

Symbol  ... 

Call up with buttons   ...  

Hexadecimal digits can be used to enter digits from 10 to 15 where possible. An example is entering the mantissa of the number in hexadecimal form (not the exponent, it is always entered in decimal form) and entering the parameters of the  and  instructions.

A special case are parameters expecting decimal form of the number. Typically number of memory register. E.g. the **RCL 2 3** instruction is stored in the program in 2 bytes, with code 43 23. During program interpretation, the register number is read by composing both digits as $2 \times 10 + 3 = 23$. If **0A** digit is used as the first digit, the number of addressable registers will be extended to 110. E.g. the entry **RCL 0A 3** will be stored with code 43 A3 and register $10 \times 10 + 3 = 103$ is used for interpretation.

10 ... 1F Letter label, A...F

Symbol **A** ... **E**, **A'** ... **E'**, **A"** ... **E"**, **F**

Call up with **A** ... **E**, **2nd A** ... **2nd E**, **2nd 2nd A** ... **2nd 2nd E**

Buttons **A** to **E** are used to quickly call up subroutines, typically with a single press. The subprogram is marked in the program with a label with the specified symbol, eg **Lbl A**. After pressing the **A** key, the relevant subprogram is called.

The buttons are available in 3 sets. The basic set, **A** to **E**, is activated by simply pressing the **A** to **E** button, without the **2nd** prefix. The second set, **A'** to **E'**, is activated by pressing **2nd** and button **A** to **E**. The third set, **A"** to **E"**, is activated by double pressing **2nd 2nd** and button **A** to **E**.

A special case is instruction **F**, with program code 1F. The code is not available directly from the keyboard as a button. It can be entered in a more complex way, using the instruction **code: 2nd 2nd INV 0 2nd 2nd 0F**. Such use would be impractical. The actual use is inside the program, to invoke subroutine using only the 1-byte instruction code.

Calling subroutines with keys **A** to **F** is performed in the same way as calling subroutine with **SBR** instruction. The address following the button code is first stored in address stack. Subroutine control is then passed. The address stack has a capacity limited to 15 subroutines.

Subroutine is terminated by **RTN** instruction. This will return from the subroutine. Original address after instruction **A** to **F** is taken from the address stack and passed control to this address. If the subroutine was

started from the keyboard, the program stops.

Subroutines can also be called from another library program. The program contains **Pgm** instruction followed by the program number and key code **A** to **F** or calling subprogram via **SBR**. If **Pgm** instruction is used in the program, the active program is not switched permanently (as it would be when used from the keyboard), the switching is only temporary for the duration of calling one subsequent subroutine. At the end of the subroutine, the control is transferred back to the original program.

Example:

RST **LRN** ... resets the pointer and activates programming mode

Lbl **A** ... creation of a temporary label

BST **code** **1** **0F** ... step back and enter correct code of the instruction F

(**CE** **+** **1** **)** **RTN** ... the subroutine increments value of register X by 1

(the **RTN** instruction is entered as **INV** **SBR**)

Lbl **A** **code** **1** **0F** **RTN** ... test program calls subroutine F

LRN ... deactivation of programming mode

1 **A** **[2]** **A** **[3]** **A** **[4]** ... test, each press of **A** increases X by 1

20 Turn off the calculator, OFF

Symbol **OFF**

Call up with buttons **2nd** **CLR**

The **OFF** instruction shuts down the calculator and enters sleep mode. The calculator shuts down in the same way after a certain period of inactivity. The idle time can be set with the **Op** **88** operation (default 30 seconds). Re-activating is possible by pressing the **CLR** button.

During the shutdown of the calculator, the RAM memory is preserved, which contains data registers, working registers (X, Y, T) and pending operations. The condition of storage is that the battery is not removed from

the calculator for more than a few minutes and that the **CLR** button is not pressed. **CLR** button would otherwise turn on the calculator but without the battery it is reset. The battery can be replaced while the calculator is turned off. If the battery is removed from the turned on calculator and no external power supply is connected, the calculator will reset when the new battery is inserted and contents of the RAM memory will be lost (contents of the registers will be set to zero).

The battery should not be removed from the calculator during EEPROM write operations - that is, during user program programming (eg, ongoing **Ins** operation). In this case, the contents of the memory may be damaged.

If the calculator is powered by an external source (USB power connector), the display will remain lit even when the calculator is turned off.

21 Alternative function, 2nd

Symbol **2nd**

Call up with the key **2nd**

The **2nd** button is used to change the meaning of the next button to an alternative function. Most buttons have 2 alternative functions. After pressing the **2nd** button once, the first alternative function is performed. After pressing the **2nd 2nd** button twice, the second alternative function is performed. Triple pressing **2nd 2nd 2nd** will return to basic functions.

Code of the **2nd** button is not stored in the program. An alternative code for the next button is always stored.

Example:

2 ln x [.6931...] ... calculates natural logarithm of the number

2 2nd ln x [.3010...] ... decimal logarithm of the number (**log** instruction)

2 2nd 2nd ln x [1] ... binary logarithm of the number (instruction **log2**)

22 Inversion of a function, INV

Symbol **INV**

Call up with the key **INV**

The **INV** button, pressed before another button code, invokes the inverse function of many buttons. In some cases, this is an additional alternative function.

The button code is stored in the program as byte 22. Most instructions either directly serve the **INV** prefix or at least reset it. In case of **Lbl** labels, the state of the **INV** prefix is preserved. This can be used to distinguish function of the program by using **INV** instruction before the program label.

Example:

RST **LRN** ... activates programming mode

Lbl **B** ... label of first subroutine

INV ... following instruction after **Lbl** will be inverted

Lbl **A** ... label of second subroutine

sin ... calculates the sine

RTN ... return from subroutine (entered with **INV** **SBR** keys)

LRN ... return to execution mode

3 **0** **A** [0.5] **B** [30] ... Subroutine **A** calculates sine of the specified angle, subroutine **B** converts the value back to the angle.

23 Natural logarithm and exponent, ln x

Symbol **ln x**

Call up with the key **ln x**

The **ln x** key calculates natural logarithm of the number on the display.

The natural logarithm uses the Euler constant with the value 2.718281828459 as the basis. If the **INV** key is pressed first, the inverse function, the natural exponent, is performed.

The argument of the function **ln x** must be a positive, non-zero number. In the case of zero, the display flashes with the value 9.9999+9999 as an error indicator. For a negative number, the absolute value of the number is calculated and the display flashes again with an error indication.

The argument of the function **INV ln x** can be both a positive and a negative number, in the range of about -23025 to +23025. A number outside this range will cause the data to overflow and indicate an error.

Example:

5 **ln x** [1.6094...] ... calculates natural logarithm of number 5 (ie 1.6094 ...)

INV **ln x** [5] ... calculates natural exponent of number on display (ie 5)

24 Clear error, CE

Symbol **CE**

Call up with the key **CE**

The **CE** key can be used to cancel the error indication E, which is performed by flashing of the display. Along with resetting the error indication, the working registers X, T and Y are corrected so that they do not contain a number with an overflow indication - ie indicated number 9.9999+9999 changes to a value of approximately 7.0773+9863, which is the maximum displayable value of the calculator.

While editing the number on the display, the last character of the entered number is deleted with the **CE** key. If the mantissa is being edited, the last character of the mantissa is deleted. If the exponent is being edited, the last character of the exponent is deleted. If an exponent with a value of 0 is deleted, the exponent is canceled and mantissa starts be edited.

25 Clear display, CLR

Symbol **CLR**

Call up with the key **CLR**

The **CLR** button performs several initialization operations. Resets started arithmetic operations, resets the error indication, turns off text mode of the display (activated by the **Op 1F** command), returns the default display fonts, turns off the **EE** exponent mode, clears the X register, and starts editing a new number with a default value of 0.

The button does not reset the T register, data registers or HIR registers.

A special function of the button is to turn on the calculator from the off state. The button is also used to turn off the calculator if the **2nd** prefix key (**OFF** function) is pressed before pressing it.

26 Subroutine with indirect address, SBR Ind

Symbol **SBR Ind**

Call up with buttons **SBR 2nd y^x**

The **SBR Ind** command is followed by a register number containing a jump address or a label symbol. The command calls a subroutine with an address or label contained in the registry. The method of addressing has been described in the chapter [Indirect Addressing](#).

Otherwise, the command works in the same way as the **SBR** command (see [71 Subroutine, SBR](#)), ie it stores current address in program stack (max. stack depth is 15 addresses) and continues at original stored address after the subroutine is terminated (termination of subroutine with the **RTN** instruction [92 Return from subroutine, RTN](#)).

27 Indirect internal instruction, HIR Ind

Symbol **HIR Ind**

Call up with buttons **2nd INV 2nd y^x** (only in the program)

The **HIR Ind** instruction works similarly to the **HIR** instruction, but instead of the working HIR register, it uses a data register whose index is contained in the HIR register specified as the instruction parameter.

*Note: For internal reasons, the **HIR Ind** function cannot be called directly from the keyboard with **HIR Ind** sequence. The function can only be used inside the program in this manner. It is possible to call up the function from the keyboard by entering its code using the **code** command (ie by pressing **2nd 2nd INV 27**).*

Example:

4 5 STO 01 ... storing test pattern 45 into data register R01

1 HIR 05 ... storing value 1 into HIR register H5

code 27 15 [45] ... Calling **HIR Ind** function with code 27. The function reads value 1 from register H5 and reads contents 45 from register R01.

28 Decimal logarithm and exponent, log

Symbol **log**

Call up with buttons **2nd ln x**

The **log** key calculates decimal logarithm of the number on the display. The decimal logarithm uses number 10 as its basis. If the **INV** key is pressed first, the inverse function, the decimal exponent, is performed.

Argument of the **log** function must be a positive, non-zero number. In case of zero, the display flashes with the value 9.9999+9999 as an error indication. For a negative number, the absolute value of the number is

calculated and the display flashes again with an error indication.

Argument of the **INV log** function can be both a positive and a negative number, in the range of about -9863 to +9863. A number outside this range will cause the data to overflow and indicate an error.

Example:

5 log [.69897...] ... decimal logarithm of number 5 (ie 0.69897...)

INV log [5] ... decimal exponent of number on display (ie 5)

29 Clear program and register T, CP

Symbol **CP**

Call up with buttons **2nd CE**

The **CP** button has two functions - deleting the user program and clearing the T register.

If the **CP** key is called from the keyboard, the user program is deleted. You will be asked to confirm the operation before deleting. Press button **1** to confirm the operation and user program will be deleted. The T register is deleted even if the program delete operation is not confirmed. In such case, the command is used only to clear the T register.

If the **CP** key is called from the program, only the T register is cleared.

2B Enter instruction code, code

Symbol **code**

Call up with buttons **2nd 2nd INV**

Use the **code** instruction to enter the direct numeric code of the button. This allows you to enter key codes that are either difficult or not at all accessible from the keyboard. The two-digit key code is entered as the

instruction parameter. The key code is executed as if the key had been pressed on the keyboard. In this way, non-standard button codes can also be entered into the program during programming. The instruction code 2B itself is ignored during program run.

Example:

5 **STO** **01** ... storing test value 5 into register R01

CLR **[0]** ... clear the display

code **43** **01** **[5]** ... calling the **RCL** button code and executing the **RCL** **01** instruction (returns content 5)

2C Binary logarithm and exponent, log2

Symbol **log2**

Call up with buttons **2nd** **2nd** **ln x**

The **log2** key calculates binary logarithm of the number on the display. The binary logarithm uses number 2 as its basis. If the **INV** key is pressed first, the inverse function, the binary exponent, is performed.

Argument of the **log2** function must be a positive, non-zero number. In the case of zero, the display flashes with the value 9.9999+9999 as an error indication. For a negative number, absolute value of the number is calculated and the display flashes again with an error indication.

Argument of the **INV** **log2** function can be both a positive and a negative number, in the range -32767 to +32767. A number outside this range will cause the data to overflow and indicate an error.

Example:

5 **log2** **[2.3219...]** ... calculates binary logarithm of number 5 (ie 2.3219 ...)

INV **log2** **[5]** ... calculates binary exponent of number on the display (ie 5)

2D Random number generator, rand

Symbol **rand**

Call up with buttons **2nd** **2nd** **CE**

The **rand** function calculates random number in the range 0 to 1 (or to the specified boundary number, with **INV** prefix), including the value 0, but excluding the value 1. LCG generator (Linear Congruential Generator) with formula $\text{RandSeed} = \text{RandSeed} * 214013 + 2531011$ is used to calculate the random number. Seed of generator has a range of 32 bits. Generated number is converted to a float by dividing by 2^{32} . This ensures that resulting random number is in the range 0 to 1, including zero, but excluding the value 1. Limited number of digits of base number (9 and half digits, number range 0 to 4294967295) ensures such granularity of the data that even after multiplication does not overflow to limit value 1.

If the prefix **INV** is pressed before the **rand** instruction, the generated random number is multiplied by the value of register X. This allows generate numbers in given range, from zero to the specified maximum number, but excluding the specified maximum number.

The random number generator counts continuously each time it is used, and this ensures that the generated sequences of numbers are not repeated. In addition, each time the calculator is reset, seed of generator is read from the EEPROM and a new value is stored. This ensures that the generated sequences are not repeated even after resetting the calculator (or after removing the battery).

By non-repetition is meant a sequence of small generated numbers. If large numbers covering the generator range (9.5 digits) are generated, the generated sequence will be repeated after a some long time.

The seed of generator is controlled by **Op** **51** and **Op** **52**.

Example of a dice roll program:

Lbl **A** **(** **6** **INV** **rand** **Int** **+** **1** **)** **RTN** ... pressing **A** generates numbers 1 to 6

30 Tangent, tan

Symbol **tan**

Call up with buttons **2nd** **1/x**

The **tan** function calculates the tangent of an angle. The angle is specified in the units set by the **Deg**, **Rad**, or **Grad** switches. If an angle calculated in radians is available, it can be converted to the current angular measure by the **Op 73** function.

Entering prefix **INV** before the **tan** instruction performs opposite function - arc tangent. The result is an angle in the currently set angular measure. If you need to convert the result to radians, you can use the **Op 72** function.

Example:

50 tan [1.19175...] ... tangent of angle 50° is value 1.19175...

99999999 INV tan [90] ... arc tangent of large number is 90°


Note:

Calculations of the arc tangent of angles around $+90^\circ$ and -90° already fall outside the range of the calculator, and thus may result be in reduced accuracy. E.g. in the above example 99999999 **INV** **tan**, the result shown is 90° , although the correct value should be approximately 89.999999427° .

31 Programming, LRN


Symbol **LRN**


Call up with the key **LRN**




The  button activates or deactivates programming mode. The programming mode was described in more detail in the [Programming](#) section.





32 Exchange of registers X and T, $x \leftrightarrow t$

Symbol 


Call up with the key 


The  key can be used to swap registers X and T. Register X is a working register or also the contents of the display. Register T is an auxiliary register (temporary). It is used to compare numbers, to convert polar and Cartesian coordinates and to calculate complex numbers.


The X register is reset by the  and  keys. The T register is reset by the  button (attention, it is also used to delete contents of user program).

The T register is not normally visible on the display. Using the   instruction, it is possible to turn on a special display mode, where the T register is displayed on the top row of the display. Switching back to the standard display format with a row of switches is possible with the instruction  .


33 Square of the number, x^2


Symbol 


Call up with the key 

The  key calculates the square of the number, that is, the multiple of the number itself.

34 Square root of the number, \sqrt{x}


Symbol 


Call up with the key 


Use the  key to calculate the square root of the number. The number must not be negative. If a negative number is calculated, the square root of

the absolute value of the number is calculated and error indication E is set (display flashes).

35 Reciprocal of the number, 1/x



Symbol 


Call up with the key 

Use the  button to calculate reciprocal of the number. If the number is zero, the value 9.9999+9999 is displayed and error indication E is set (display flashes). This is often used in programs to turn on error indication and to indicate program malfunction.


36 Library program selection, Pgm


Symbol 

Call up with buttons  

The  button can be used to select program from the library. Parameter is two-digit number of the selected program, starting with the number 01. The number 00 activates the main user program. After selecting a program, the display briefly shows the name of the library, the number of the selected program and the length of the program in bytes. The number of programs in the library is 50. When selecting a program out of range, the error indication E flashes.

If a program from the library is selected, subroutines from that program will be run.

After activating the programming mode with the  button, the content of the selected library program is displayed. You can browse and view the program, but you cannot edit it. The library program is marked with an asterisk between the address and the current byte (read only flag).

Selecting program 00 selects the user program ("Main Program" appears briefly on the display). Only the user program can be edited. The  button has a similar function, it also switches the selection to the user

program.

By placing the prefix **INV** in front of the **Pgm** key, the working register X (ie contents of the display) is set to the value of the number of the selected library program and the display flashes briefly the data about the selected program, similar to switching the program. In this way, it is possible to programmatically test which program is currently active.

If the **Pgm** function is used inside a running program, the program will not be switched permanently. The switch only applies to the following subroutine call instruction (letters **A** to **F** or **SBR** subroutine). In this way, you can call a program from another library program, or even from the main user program.

By entering the code **Ind**, the function can be changed to indirect addressing **Pgm Ind** (see [62 Indirect selection of library program, Pgm Ind](#)).

37 Conversion of Cartesian and polar coordinates, P->R

Symbol **P->R**

Call up with buttons **2nd** **x<>t**

The **P->R** key converts the coordinates from polar expression to Cartesian coordinates. Prior to the operation, the T register (ie the auxiliary register) contains the radius and the X register (display content) contains the angle. The angle is specified in the currently selected angular measure (**Deg**, **Rad** and **Grad** buttons). After the operation, the T register (auxiliary register) contains the X coordinate, the X register (display content) contains the Y coordinate. The angle can be converted from radians to currently selected angular measure by the **Op 73** instruction before the operation.

By putting the prefix **INV** before the instruction **P->R**, the opposite operation is performed, the conversion of Cartesian coordinates to polar. Before the operation, the T register contains the X coordinate, the X register contains the Y coordinate. After the operation, the T register contains the radius and the X register contains the angle. The angle is specified in the currently selected angular measure. If it is necessary to convert the angle to radians after the operation, this can be done with the

instruction **Op** **72**.

The **Op** **1E** instruction can be used to activate a special display mode in which the content of the T register is displayed on the first row of the display. This can facilitate the use of coordinate conversion. Use the **Op** **1D** instruction to restore the display mode.

Example.

1 **0** **x<>t** ... entering radius 10 in register T

3 **0** ... entering angle of 30 ° in register X

P->R [5] ... conversion of polar coordinates to Cartesian. The display shows coordinate Y = 5

x<>t [8.6602...] ... switching X and T displays coordinate X = 8.6602...

x<>t [5] ... switching registers X and T back

INV **P->R** [30] ... backward recalculation, angle of 30 ° is displayed

x<>t [10] ... display register T with radius 10

38 Sine, sin

Symbol **sin**

Call up with buttons **2nd** **x^2**

The **sin** function calculates the sine of the angle. The angle is specified in the units set by the **Deg**, **Rad**, or **Grad** switches. If an angle calculated in radians is available, it can be converted to current angular measure by the **Op** **73** function.

Entering the **INV** prefix before the **sin** instruction performs the opposite function - arc sine. The result is an angle in the currently set angular measure. If you need to convert the result to radians, you can use the **Op** **72** function.

The angle calculated by the arc sine function is in the range -90° to + 90°.

The input value of the arc sine function must be in the range -1 to +1. If it is outside the specified range, the value is limited to the valid range and error E is indicated (display flashes).

39 Cosine, cos

Symbol **cos**

Call up with buttons **2nd** **Vx**

The **cos** function calculates the cosine of an angle. The angle is specified in the units set by the **Deg**, **Rad**, or **Grad** switches. If an angle calculated in radians is available, it can be converted to the current angular measure by the **Op** **73** function.

Entering the prefix **INV** before the **cos** instruction performs the opposite function - arc cosine. The result is an angle in the currently set angular measure. If you need to convert the result to radians, you can use the **Op** **72** function.

The angle calculated by the arc cosine function is in the range 0° to +180°. The input value of the arc cosine function must be in the range -1 to +1. If it is outside the specified range, the value is limited to the valid range and error E is indicated (display flashes).

3A Temperature, Temp

Symbol **Temp**

Call up with buttons **2nd** **2nd** **LRN**

The **Temp** button can be used to determine the temperature of the processor chip, and thus the room temperature (the processor practically does not heat up due to the low frequency). The resolution of the temperature measurement is 1° C. The temperature data is very inaccurate, even after calibration it may differ by several degrees, and is only indicative.

The temperature measurement must first be calibrated before first use.

Without calibration, the value eg 100 is displayed. To calibrate, enter the current room temperature in whole degrees into the calculator, eg 24. Press **INV** **Temp** to save the set value in the EEPROM memory as the difference between the actual and measured temperature. From then on, the difference stored in the EEPROM memory will be added to the measured data. As a rule, a temperature measurement deviation of up to 2° C can be achieved, but only at temperatures close to the calibration temperature. At more distant temperatures, the deviation may be larger. The temperature calibration can be repeated at any time using the same procedure.

3B Exchange of X and Y registers, x<>y

Symbol **x<>y**

Call up with buttons **2nd** **2nd** **x<>y**

While entering arithmetic operations, the display contains the value of the working register X. If the calculation is performed with two registers (eg addition instructions), the arithmetic operation stack contains the second register, Y. The **x<>y** key can be used to swap registers of the operation. This may be necessary for operations with an important order of operands, such as division or subtraction.

The **x<>y** key can also be used while repeating last calculation performed with the **=** key. In this case, the second entered parameter is stored in register Y, the first operand of the operation is entered in register X (display data).

Example:

3 **y^x** **2** = [9] ... calculates value $3^2 = 9$

5 **=** [25] ... calculates value $5^2 = 25$

0 **.** **5** **x<>y** [2] ... stores the value 0.5 in the Y register

9 **=** [3] ... calculates value $9^{0.5} = 3$

3C Hyperbolic sine, sinh

Symbol **sinh**

Call up with buttons **2nd** **2nd** **x^2**

The **sinh** key is used to calculate the hyperbolic sine function.

By putting the prefix **INV** before the **sinh** instruction, an inverse operation, hyperbolic arc sine, is performed.

Example:

1 **sinh** [1.1752...] ... calculation of $\sinh(1) = 1.1752...$

INV **sinh** [1] ... backward calculation of $\operatorname{asinh}(1.1752...) = 1$

3D Hyperbolic cosine, cosh

Symbol **cosh**

Call up with buttons **2nd** **2nd** **Vx**

The **cosh** key is used to calculate the hyperbolic cosine function.

By putting the prefix **INV** before the **cosh** instruction, an inverse operation, hyperbolic arc cosine, is performed.

Example:

1 **cosh** [1.5430...] ... calculation of $\cosh(1) = 1.5430...$

INV **cosh** [1] ... backward calculation of $\operatorname{acosh}(1.5430...) = 1$

3E Hyperbolic tangent, tanh

Symbol **tanh**

Call up with buttons **2nd** **2nd** **1/x**

The **tanh** key is used to calculate the hyperbolic tangent function.

By putting the prefix **INV** before the **tanh** instruction, an inverse operation, the hyperbolic arc tangent, is performed.

Example:

1 **tanh** [.76159...] ... calculation of $\tanh(1) = 0.76159...$

INV **tanh** [1] ... backward calculation of $\operatorname{atanh}(0.76159...) = 1$

40 Indirect addressing, Ind

Symbol **Ind**

Call up with buttons **2nd** **Ind**

The **Ind** (Indirect) button is used to enter indirect parameters of the operation, where the required data is not taken from the instruction code, but from the content of the data register. For more information, see the chapter [Indirect Addressing](#).

41 Program step forward, SST

Symbol **SST**

Call up with the key **SST**

The **SST** (Single Step) key increases the program address pointer by 1 in programming mode.

In execution mode, 1 program instruction is executed, making it possible to step through the program for debugging. If program stepping is combined with subroutine execution, the correct return from subroutines may not occur (the calculator will not remember the return address from the

subroutine).

For more information, see the [Programming](#) chapter.

42 Store number to data register, STO

Symbol **STO**

Call up with the key **STO**

Use the **STO** (Store) button to store the number on the display in the data register. Two-digit code of the register number, from 00 to 99, is entered as an instruction parameter.

By entering the code **Ind** after the **STO** instruction, but before entering the parameter, the **STO** instruction changes to an indirect **STO Ind** instruction (with code 72, see [72 Indirect store number to data register, STO Ind](#)). For more information on indirect addressing, see the chapter [Indirect Addressing](#).

43 Retrieving number from data register, RCL

Symbol **RCL**


Call up with the key **RCL**


Use the **RCL** (Recall) key to recall number from the data register on the display. Two-digit register number code, from 00 to 99, is entered as an instruction parameter.






By entering the **Ind** code after the **RCL** instruction, but before entering the parameter, the **RCL** instruction changes to an indirect **RCL Ind** instruction (with code 73, see [73 Indirect retrieving number from data register, RCL Ind](#)). For more information on indirect addressing, see the chapter [Indirect Addressing](#).



44 Add and subtract number from data register, SUM

Symbol 

Call up with the key 


Using the  (Summation) key, the number on the display can be added to the data register. Two-digit code of register number, from 00 to 99, is entered as instruction parameter.



By entering the code  after the  instruction, but before entering the parameter, the  instruction changes to indirect   instruction (with code 74, see [74 Indirect add and subtract number from data register, SUM Ind](#)). For more information on indirect addressing, see the chapter [Indirect Addressing](#).

By putting the prefix  before the  instruction, the opposite function is performed - subtracting the number from the data register.

45 Power and root, y^x






Symbol 

Call up with the key 


The instruction  powers the number Y (first operand, in the operation stack) by the number X (second operand, number on the display). If the  prefix is pressed first, an inverse operation, the root, is performed.

The first operand Y should normally be a non-negative number. Only if the exponent X is an integer, then first operand can be a negative number. In this case, the result is negative if the exponent was an odd number.


Example:

     [-2187] ... power $(-3)^7 = -2187$

46 Insert empty byte into program, Ins

Symbol 

Call up with buttons  

The  (Insert) key, pressed in programming mode, inserts an empty byte with the value FF at the current program position. The following data is moved away, up to end of memory or up to next empty FF byte. Absolute addresses are not recalculated when moved.


See the [Programming](#) chapter for more information.


Note that some instructions allow you to enter a parameter in the HEX code and insert valid byte with value of 0FF into the code. If possible, avoid such a byte value, as it would be interpreted as empty space and damaged during program moving in memory.

47 Clearing data registers, CMs

Symbol 

Call up with buttons  


The  button can be used to clear contents of all data registers (it does not apply to HIR registers).

If the prefix  is pressed before calling the function, only the data registers R01 to R06 (registers used for statistic functions), register T and register X are cleared.

48 Exchange number with data register, Exc

Symbol 

Call up with buttons  

Use the  (Exchange) button to exchange number on the display with content of the data register. A two-digit code of register number, from 00 to

99, is entered as instruction parameter.

By entering the **Ind** code after the **Exc** instruction, but before entering the parameter, the **Exc** instruction changes to an indirect **Exc Ind** instruction (with code 63, see [63 Indirect exchange number with data register, Exc Ind](#)). For more information on indirect addressing, see the chapter [Indirect Addressing](#).

49 Multiply and divide data register, Prd

Symbol **Prd**

Call up with the key **2nd SUM**

Use the **Prd** (Product) button to multiply data register by the number on the display. Two-digit code of the register number, from 00 to 99, is entered as an instruction parameter.

By entering the code **Ind** after the **Prd** instruction, but before entering the parameter, the **Prd** instruction changes to indirect **Prd Ind** instruction (with code 64, see [64 Indirect multiply and divide data register, Prd Ind](#)). For more information on indirect addressing, see the chapter [Indirect Addressing](#).

By entering the prefix **INV** before the **Prd** instruction, the opposite function is performed - dividing data register by the number on the display.

4A Battery voltage detection, Bat

Symbol **Bat**

Call up with buttons **2nd 2nd SST**

The **Bat** button can be used to determine voltage of the battery. The voltage is displayed in volts. Battery discharge status can be assessed from the battery voltage. The fresh CR2032 battery has a voltage of around 3V. The calculator is able to work from a minimum voltage of about 2.5V. When powered from the USB connector, the indicated supply voltage will

be around 2.9V (a stabilizer is used to reduce the voltage).

Entering the **INV** prefix before pressing **Bat** will display the battery voltage as a percentage. The value 0 to 100% corresponds to a battery voltage of 2.4V to 2.9V.

Accuracy of battery voltage measurement data is not guaranteed, it is for guidance only.

4B Factorial, x!

Symbol **x!**

Call up with buttons **2nd** **2nd** **STO**

Button **x!** can be used to calculate the factorial. Factorial is a number that is created by multiplying the values 1, 2, 3, ... up to the entered input number x. The calculator calculates the factorial using the approximation function (Stieltjes Gamma function). This allows fast and accurate calculations, including decimal factors.

Input number must not be negative. Maximum value of the factorial that the calculator can calculate is 3208 (result 8.61680144+9856).

By entering the prefix **INV** before pressing **x!**, repeated integer multiplication is used instead of calculating the approximation function. The input can be an integer in the range 0 to 3209. This method is not more accurate or faster than the calculation using the approximation function, it only serves as a reference value.

Example:

6 **x!** [720] ... factorial of number 6! = 1*2*3*4*5*6 = 720

INV **6** **x!** [720] ... reference factorial by multiplication 6! = 720

6 **|** **1** **x!** [868.9568...] ... decimal factorial 6.1! = 868.9568...

4C Natural logarithm of factorial, ln x!

Symbol **ln x!**

Call up with buttons **2nd** **2nd** **RCL**

The **ln x!** button calculates natural logarithm of the factorial. The input value is a non-negative number, including decimal numbers. Unlike the function for calculating the factorial **n!**, this function is not so limited by the range of the calculator. The input can be almost any positive number.

Example:

6 **ln x!** [6.57925...] ... natural logarithm $\ln 6! = 6.57925...$

6 **n!** **ln x** [6.57925...] ... control calculation

4D Decimal logarithm of factorial, $\log x!$

Symbol **log x!**

Call up with buttons **2nd** **2nd** **SUM**

The **log x!** button calculates decimal logarithm of the factorial. The input value is a non-negative number, including decimal numbers. Unlike the function for calculating the factorial **n!**, this function is not so limited by the range of the calculator. The input can be almost any positive number. The function can also be used to count very large factorials.

Example 1:

6 **log x!** [2.85733...] ... decimal logarithm $\log 6! = 2.85733...$

6 **n!** **log x** [2.85733...] ... control calculation

Example 2, 123456789!:

1 **2** **3** **4** **5** **6** **7** **8** **9** **log x!** [945335859.45538] ... logarithm of factorial

INV **Int** **INV** **log** [2.8535...] ... exponent of decimal part

... result $123456789! = 2.85351252... \cdot 10^{945335859}$

Note: According to the calculator, the result is $123456789! = 2.8535125217299 \cdot 10^{945335859}$. The correct value should be $2.8535125219128 \cdot 10^{945335859}$. Mantis of the result matches 10 digits. This is given by the use of logarithm. Internal accuracy of the calculator is 19 digits. With such accuracy the calculator calculates the logarithm of the factorial. Integer part of the logarithm represents 9 digits of the exponent ($10^{945335859}$). After removing 9 digits of integer part, mantissa with an accuracy of 10 digits remains, which is the achievable accuracy of the result mantissa. When dividing the logarithm into a part of the exponent and the mantissa, it is necessary to take into account the reduction of accuracy of mantissa of the result.

4E Modulo floor, mod2

Symbol **mod2**

Call up with buttons **2nd** **2nd** **y^x**

The modulo operation divides the first operand Y (in the stack) by the second operand X (on the display), converts the result to an integer, multiplies the second operand X by it, and subtracts it from the first operand Y. The result is the remainder after division.

The **mod2** instruction is similar to the **mod** instruction (see [5E Modulo trunc, mod](#)) and gives the same result for positive numbers. The difference is reflected in the negative numbers. The **mod2** operation uses the floor function to round the result, ie rounding down. The result has the same sign as the second operand (unlike the **mod** function, which retains the sign of the first operand).

The **mod2** instruction can be used, for example, to normalize an angle to the range 0 to 359°, because it also treats negative angles correctly.

Example:

$2 \div 2 \text{ mod } 2 \ 0.5 = [0.2] \dots 2.2 \text{ mod } 2 \ 0.5 = 0.2$

$2 \div 2 \ +/- \text{ mod } 2 \ 0.5 = [0.3] \dots -2.2 \text{ mod } 2 \ 0.5 = 0.3$

$2 \div 2 \text{ mod } 2 \ 0.5 \ +/- = [-0.3] \dots 2.2 \text{ mod } 2 \ -0.5 = -0.3$

$2 \div 2 \ +/- \text{ mod } 2 \ 0.5 \ +/- = [-0.2] \dots -2.2 \text{ mod } 2 \ -0.5 = -0.2$

50 Absolute value, $|x|$

Symbol $|x|$

Call up with buttons $2\text{nd} \ |x|$

The $|x|$ function adjusts the number to an absolute value (removes negative sign of the number).

If the prefix INV is given before pressing $|x|$, a sign operation is performed with the content of register X (number on the display). If the content of the register is less than 0, the operation will result in the number -1. If the content of the register is greater than 0, the result will be +1. If the content of the register is 0, 0 remains.

Instead of $\text{INV} \ |x|$, the $\text{Op} \ 10$ operation can also be used.

51 Step the program back, BST


Symbol BST


Call up with the key BST


The BST (Back Step) key decreases the program address pointer by 1 when in programming mode.

For more information, see chapter [Programming](#).




52 Exponent mode, EE

Symbol 









Call up with the key 

Press the  button to switch on the exponent mode. If the key is pressed while entering a number, the exponent will be entered. At the same time, the display mode in scientific notation with the exponent is switched on.


If the button is pressed outside the number editing, the display mode in scientific notation with the exponent is switched on and the editing of the number exponent is started. This function is often used to remove hidden digits of the number, because when you start editing, only the displayed digits are loaded into the editor, not the full exact value of the number.

Pressing the  prefix before pressing  exits the exponent display mode. Another way to exit exponent display mode is to press the  key.

Example:

        [3.1416] ... rounds number to 4 places

53 Left parenthesis, (


Symbol 


Call up with the key 

Button  starts calculating part of the expression.


54 Right parenthesis,)

Symbol 



Call up with the key 

The  button ends the calculation of part of the expression.


55 Division, :

Symbol 


Call up with the key 

Button  divides the first operand by the second operand. If this is the lowest level of the expression, the calculation can be repeated for another first operand by pressing  repeatedly.

56 Delete a byte from the program, Del

Symbol 



Call up with buttons  


The  (Delete) key pressed in programming mode deletes a byte from the current program position. The following data are pulled up, up from the end of memory or from the next empty FF byte. Absolute addresses are not recalculated when moved.

See the chapter [Programming](#) for more information.

57 Technical mode, Eng

Symbol 

Call up with buttons  

The  (Engineer) button activates the technical (engineering) mode of displaying the number. The number is displayed with an exponent that is a multiple of 3 (decimal and octal display mode) or 4 (hexadecimal and binary display mode).

Technical mode  is not deactivated by pressing . To turn it off, it

is necessary to use the sequence **INV Eng**.

Example:

5 **:** ... let us have a voltage of 5 V

2 **EE** **3** **+/-** ... divide the voltage by 2 mA current

= **Eng** [2.5+3] ... value of resistor will be 2.5 kOhm

58 Rounding, Fix

Symbol **Fix**

Call up with buttons **2nd** **|**

Use the **Fix** key to round the number shown on the display to the specified number of decimal places. A number representing the number of decimal places after the decimal point is entered as a parameter. The digit can be **0** to **9** (representing 0 to 9 decimal places), but also the hexadecimal digit **0A** to **0D**, which means 10 to 13 decimal places.

In rounding mode, the number is padded to the right with zeros, up to specified number of decimal places. Entering **INV Fix** or **Fix 0F** turns off the rounding. In this case, the number is displayed in full precision and the trailing insignificant zeros are removed.


The original TI-58/59 calculator used **Fix 9** sequence to turn off the rounding. This is considered a valid 9-decimal setting for the ET-58. When importing a program from the TI-58/59, it is necessary to make a correction and replace **Fix 9** with code **INV Fix** or **Fix 0F**.



Rounding only affects the display of the number. Internally, the number is still calculated in full accuracy. If hidden digits really need to be removed, this can be done with the **EE** key (see [52 Exponent mode, EE](#)).


The set rounding mode also affects the way of displaying very small numbers. If rounding is on and exponent mode is not on, the display will show zeros for small numbers, even if valid digits have passed beyond the right border of the display. If rounding is not turned on, the calculator



switches to the exponent display if the exponent is less than -3.

59 Integer number, Int

Symbol 

Call up with buttons  

The  (Integer) button can be used to remove digits after the decimal point, ie to trim the number to an integer. The function has the same meaning as rounding to zero.

If the prefix  is used before the  command, the inverse function is performed - removing the whole part of the number and leaving the decimal part (frac, fraction).

Example:

$$\boxed{2} \boxed{.} \boxed{3} \boxed{\text{Int}} [2] \dots \text{int}(2.3) = 2$$




$$\boxed{2} \boxed{.} \boxed{3} \boxed{+/-} \boxed{\text{Int}} [-2] \dots \text{int}(-2.3) = -2$$


$$\boxed{2} \boxed{.} \boxed{3} \boxed{\text{INV}} \boxed{\text{Int}} [0.3] \dots \text{frac}(2.3) = 0.3$$

$$\boxed{2} \boxed{.} \boxed{3} \boxed{+/-} \boxed{\text{INV}} \boxed{\text{Int}} [-0.3] \dots \text{frac}(-2.3) = -0.3$$

5A Adjust the display contrast, LCD

Symbol 

Call up with buttons   

Use the  button to adjust the display contrast. After pressing the button, the instruction will ask you to enter the number 0 to 9. Number 0 represents minimum contrast (characters are gray or barely visible), number 9 is maximum contrast (there are dark rectangles below the characters).

The contrast of the LCD display depends on the supply voltage. By

adjusting the contrast, it is possible to get out of the visible range. In this case, it may be necessary to adjust the display contrast memorize. If no characters are visible on the display (the calculator looks off), try setting a higher display contrast from the keyboard using a higher number. Alternatively, press **CLR** first, maybe the calculator is switched off. Conversely, if there are black rectangles on the display, set a lower value for the display contrast.

By entering the prefix **INV** before the **LCD** instruction, the currently set value of the display contrast can be found out (the value 0 to 9 is returned in register X).

5B Shift left, SHL

Symbol **SHL**

Call up with buttons **2nd** **2nd** **EE**

The **SHL** key can be used to move the first argument Y (in the operation stack) to the left by the number of bits given by the second argument X (number on the display). The 1 bit left shift operation corresponds to operation of multiplication by the number 2.

Example:

1 **2** **3** **SHL** **4** **=** [1968] ... $123 \ll 4 = 123 * 2^4 = 1968$

nebo v HEX kódu: $123 \ll 4 = 0x7B \ll 4 = 0x7B0 = 1968$

5C Shift right, SHR

Symbol **SHR**

Call up with buttons **2nd** **2nd** **(**

The **SHR** key can be used to move the first argument Y (in the operation stack) to the right by the number of bits given by the second argument X (number on the display). The 1 bit right shift operation corresponds to

operation of division by the number 2.

Example:

1 **2** **3** **SHR** **4** **=** [7.6875] ... $123 \gg 4 = 123 / 2^4 = 7.6875$

nebo v HEX kódu: $123 \gg 4 = 0x7B \gg 4 = 0x7.B = 7.6875$

5D Rounding, round

Symbol **round**

Call up with buttons **2nd** **2nd** **)**

The **round** button rounds the number on the display to the nearest whole number. If the decimal part is greater than or equal to 0.5, the number is rounded up. If the decimal part is less than 0.5, it is rounded down.

Unlike the **Fix** function, rounding is done with an actual number, not just for display.

Putting the prefix **INV** before the **round** function removes the whole part of the number using rounding of the number downwards (floor). This operation will result in a number that will always be positive and will be in range from 0 (inclusive) to 1 (exclusive). For positive numbers, the result is the same as the **INV** **Int** function (see [59 Integer number, Int](#)). For negative numbers, 1 is added to nonzero result.

Example:

2 **.** **3** **round** [2] ... $\text{round}(2.3) = 2$

2 **.** **5** **round** [3] ... $\text{round}(2.5) = 3$

2 **.** **3** **+/-** **round** [-2] ... $\text{round}(-2.3) = -2$

2 **.** **5** **+/-** **round** [-3] ... $\text{round}(-2.5) = -3$

2 **.** **6** **+/-** **round** [-3] ... $\text{round}(-2.6) = -3$

2 **.** **3** **INV** **round** [0.3] ... $2.3 - \text{floor}(2.3) = 2.3 - 2 = 0.3$

2 **.** **6** **INV** **round** [0.6] ... $2.6 - \text{floor}(2.6) = 2.6 - 2 = 0.6$

2 **.** **3** **+/-** **INV** **round** [0.7] ... $-2.3 - \text{floor}(-2.3) = -2.3 - (-3) = 0.7$

2 **.** **6** **+/-** **INV** **round** [0.4] ... $-2.6 - \text{floor}(-2.6) = -2.6 - (-3) = 0.4$

5E Modulo trunc, mod

Symbol **mod**

Call up with buttons **2nd** **2nd** **:**

Modulo operation divides the first operand Y (in the stack) by the second operand X (on the display), converts the result to an integer, multiplies the second operand X by it, and subtracts it from the first operand Y. The result is the remainder after division.

The **mod** instruction is similar to the **mod2** instruction (see [4E Modulo floor, mod2](#)) and gives the same result for positive numbers. The difference is reflected in the negative numbers. The **mod** operation uses the trunc function to round the result, that is, rounding to zero. The result has the same sign as the first operand (unlike the **mod2** function, which retains the sign of the second operand).

Example:

2 **.** **2** **mod** **0** **.** **5** **=** [0.2] ... $2.2 \bmod 0.5 = 0.2$

2 **.** **2** **+/-** **mod** **0** **.** **5** **=** [-0.2] ... $-2.2 \bmod 0.5 = -0.2$

2 **.** **2** **mod** **0** **.** **5** **+/-** **=** [0.2] ... $2.2 \bmod -0.5 = 0.2$

2 **.** **2** **+/-** **mod** **0** **.** **5** **+/-** **=** [-0.2] ... $-2.2 \bmod -0.5 = -0.2$

60 Degrees, Deg

Symbol **Deg**

Call up with buttons **2nd** **x**

The **Deg** button switches the calculations of trigonometric functions to degrees (full angle is 360°).

If calculations need to be performed independently of the selected angular measure, the **Op** **72** and **Op** **73** functions can be used for conversions.

61 Jump, GTO

Symbol **GTO**

Call up with the key **GTO**

The **GTO** (Go To) button is used to perform an unconditional jump in a program. The parameter is entered with either the 3-digit numeric code of the absolute destination address or the button representing the label in the program.

Using the **Ind** key after the **GTO** code, but before entering the jump address, the instruction changes to an instruction with indirect addressing, **GTO** **Ind** (instruction code 83, see [83 Indirect jump, GTO Ind](#)).

If the **GTO** instruction is used in execution mode, the program pointer is set to the selected address. This can be used to move the pointer during programming (see [Programming](#) for more details).

For more information on addressing methods, see chapter [Indirect Addressing](#).

The **GTO** button has another special function. If you hold down the **GTO** button while the program is running, the current content of the display (register X content) will flash in the bottom row of the display and current program address with currently executed command will flash in top row. However, this monitoring slows down the program many times over.

62 Indirect selection of a library program, Pgm Ind

Symbol **Pgm Ind**

Call up with buttons **2nd LRN 2nd y^x**

The **Pgm Ind** instruction allows you to select a library program in the same way as the **Pgm** instruction (see [36 Library program selection, Pgm](#)), only instead of from the instruction parameter the library number is taken from the data register.

The instruction is created by pressing the **Ind** key after the **Pgm** key, but before entering the parameter, which will be a 2-digit register number. For more information on indirect addressing, see the chapter [Indirect Addressing](#).

Example:

2 STO 01 ... the library number 02 is stored in register R01

Pgm Ind 01 [ML-02 (875)] ... the library program 02 is activated

63 Indirect exchange number with data register, Exc Ind

Symbol **Exc Ind**

Call up with buttons **2nd RCL 2nd y^x**

The **Exc Ind** instruction can be used to exchange the number on the display with the data register similarly to the **Exc** instruction (see [48 Exchange of number with data register, Exc](#)), only instead of the instruction parameter the register number is taken from the data register.

The instruction is created by pressing the **Ind** key after the **Exc** key, but before entering the parameter, which will be a 2-digit register number. For more information on indirect addressing, see the chapter [Indirect Addressing](#).

64 Indirect multiply and divide data register, Prd Ind

Symbol **Prd Ind**

Call up with buttons **2nd SUM 2nd y^x**

The **Prd Ind** instruction can be used to multiply or divide the contents of the data register by the number on the display in the same way as the **Prd** instruction (see [49 Multiply and divide data register, Prd](#)), only instead of the instruction parameter the register number is taken from the data register.

The instruction is created by pressing the **Ind** key after the **Prd** key, but before entering the parameter, which will be a 2-digit register number. For more information on indirect addressing, see the chapter [Indirect Addressing](#).

65 Multiplication, x

Symbol **x**

Call up with the key **x**

The **x** key multiplies the first operand by the second operand. If this is the lowest level of the expression, the calculation can be repeated for another first operand by pressing **=** repeatedly.

66 Delay, Pause

Symbol **Pause**

Call up with buttons **2nd GTO**


The **Pause** command specified in the program pauses program execution for 0.25 seconds and displays the current content of the display (content of register X).



See also monitoring the program run with the **GTO** key ([61 GTO jump](#)).



The commands **Op 80** and **Op 81** are used to briefly pause the program




without displaying the display content.

67 Equal, x=t

Symbol 

Call up with buttons  

Instrukce  umožňuje porovnat registr X (obsah displeje) s pomocným registrem T (nastaveným tlačítkem ). Jsou-li registry shodné, provede se skok na adresu, která je zadána jako parametr instrukce. Adresou může být absolutní adresa nebo návěští.








Instrukce  nemá zvláštní kód pro nepřímé adresování. Přesto nepřímé adresování umožňuje a to tak, že kód  se uloží do programu za kód . Bude-li podmínka splněna, načte se ze zadaného registru absolutní adresa nebo kód návěští, tak jak je podrobněji popsáno v kapitole [Nepřímé adresování](#).

Je-li před kódem  zadán prefix , provede se inverzní funkce - skok na zadanou adresu se provede naopak v případě neshody registrů.


Při testování shody není testována absolutní shoda celé mantisy, protože díky výpočtům se mohou čísla nepatrně lišit. Je počítáno s malou povolenou tolerancí. Po větším množství operací se může odchylka dostat mimo povolenou toleranci a shoda nebude správně vyhodnocena. Typickým případem je opakované odečítání celého čísla. Pokud je to možné, je doporučeno porovnávaná čísla zaokrouhlovat nebo testovat v intervalu s větší tolerancí.

Example nepřímého adresování:

  ... activates programming mode

       ... při shodě skočí na adresu z R01

  ... větev na adrese 7 vrací hodnotu 1

 ... návrat do prováděcího módu

7 **STO** **01** ... uložení adresy skoku '7' do registru R01

5 **x/r** **2** **A** [0] ... čísla 2 a 5 se neshodují, výsledkem je hodnota 0

5 **A** [1] ... čísla 5 a 5 se shodují, je navrácen kód 1

68 Žádná operace Nop

Symbol **Nop**

Call up with buttons **2nd** **8**

Příkaz **Nop** (No Operation) je prázdný příkaz, neprovádějící žádnou operaci. Slouží pouze k zaplnění nevyužitého místa v programu. Podobnou funkci mají i mnohé jiné kódy programu, např. kódy A0 až FE, a proto jsou vhodné např. jako návěští programu.

69 Speciální operace Op

Symbol **Op**

Call up with buttons **2nd** **9**

Instrukce **Op** zajišťuje provedení speciálního příkazu kalkulátoru. Za instrukcí následuje 1-bajtový kód parametru.

Doplněním kódu **Ind** za kódem **Op**, ale ještě před uvedením parametru, se instrukce změní na instrukci s nepřímým adresováním, **Op Ind**, s kódem 84 (viz [84 Nepřímá speciální operace Op Ind](#)). V případě nepřímého adresování se parametr instrukce nečte z programu, ale z datového registru, jehož číslo je uvedeno jako parametr operace.

Instrukce **Op** je rozsáhlá instrukce s mnoha operacemi, a proto je uvedena v samostatné kapitole, [Speciální operace Op](#).

6A Relativní skok REL

Symbol **REL**

Call up with buttons **2nd** **2nd** **GTO**

Instrukce **REL** provede relativní skok podle hodnoty následujícího parametru. Parametrem instrukce je dekadické číslo 00 až 99. Hodnota parametru se přičte k adrese následující za instrukcí **REL** a provede se skok na danou adresu. Např. parametr 01 znamená přeskočení 1 následujícího bajtu, parametr 00 znamená pokračování v programu bez provedení skoku.

Je-li před instrukcí **REL** použit prefix **INV**, provede se skok směrem zpět. Hodnota parametru se odečte od adresy následující za instrukcí **REL**. Např. parametr 03 znamená skok zpět na začátek instrukce **INV REL 03**, parametr 00 znamená pokračování v programu bez provedení skoku.

Relativní skok **REL** funguje podobně jako absolutní adresování, tj. provede se rychlý skok bez potřeby používat návěští. Přitom je adresa skoku nezávislá na poloze v paměti, kód lze v programu libovolně posouvat. Příkaz **REL** je vhodný k použití při krátkých skocích uvnitř programu.

Example:

RST **LRN** ... activates programming mode

Lbl **A** **+** **1** **=** **Pause** **INV** **REL** **0** **7**

LRN ... návrat do prováděcího módu

- Po stisku **A** program cykluje a opakovaně zvyšuje číslo na displeji.

6B Nepřímá inkrementace/dekrementace registru Inc Ind

Symbol **Inc** **Ind**

Call up with buttons **2nd** **2nd** **.** **2nd** **y^x**

Instrukce **Inc** **Ind** inkrementuje/dekrementuje obsah datového registru stejně jako instrukce **Inc** (viz [9C Inkrementace a dekrementace registru Inc](#)), jen namísto z parametru instrukce se číslo registru převezme

z datového registru.

Instrukce se vytvoří stiskem klávesy **Ind** za klávesou **Inc**, ale ještě před zadáním parametru, kterým bude 2-místné číslo registru. Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

6C Nepřímá operace s registry Reg Ind

Symbol **Reg Ind**

Call up with buttons **2nd 2nd RST** nn **2nd y^x**

Instrukce **Reg Ind** provede operaci mezi registry stejně jako instrukce **Reg** (viz [8A Operace s registry Reg](#)), jen namísto z druhého parametru instrukce se číslo zdrojového registru převezme z datového registru. Cílový registr není kódem **Ind** změněn, je určen prvním parametrem instrukce **Reg**.

Instrukce se vytvoří stiskem klávesy **Ind** za prvním parametrem instrukce **Reg**, ale ještě před zadáním druhého parametru (pozor, **Ind** nelze uvést ihned za instrukcí **Reg**). První parametr určuje cílový operand a kód operace a je shodný pro instrukci **Reg** i pro instrukci **Reg Ind**. Druhým operandem instrukce **Reg** je číslo registru použitého jako zdrojový operand. U instrukce **Reg Ind** je druhým parametrem číslo registru obsahující číslo registru zdrojového operandu. Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

6D Nepřímá podmínka IF Ind

Symbol **IF Ind**

Call up with buttons **INV 2nd 2nd SBR**

Instrukce **IF Ind** provede podmíněný skok stejně jako instrukce **IF** (viz [7A Podmíněný skok IF](#)), jen namísto přímého porovnání datových registrů se z kódu instrukce převezmou indexy registrů, načtou se jejich obsahy a použijí se jako indexy datových registrů k porovnání. Nepřímé adresování se uplatní pro oba operandy, první i druhý. Nelze jeden z registrů adresovat

přímo a druhý nepřímě. Adresa skoku může být nepřímá - v tom případě se kód **Ind** použije před zadáním adresy skoku (před 3. operandem instrukce **IF**).

Instrukce **IF Ind** se vytváří poněkud nestandardním postupem oproti jiným nepřímým instrukcím (protože kód **Ind** má zde význam platného parametru). Nejdříve se stiskne prefix **INV** a poté klávesa **IF**. To zajistí, že se namísto kódu **IF** uloží kód **IF Ind**. Následující parametry se zadají stejně jako u instrukce **IF**. Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

6E Bitový součin AND

Symbol **AND**

Call up with buttons **2nd 2nd x**

Instrukce **AND** provede bitovou operaci AND (bitový součin) mezi prvním a druhým operandem. Jedná-li se o nejnižší úroveň výrazu, může být výpočet opakován pro jiný první operand opakováním stiskem **=**.

Bitový součin znamená, že výsledkem operace je bit '1' pouze v případě, kdy mají oba vstupní bity hodnotu '1'.

Example:

BIN ... přepnutí na bitový mód zobrazení

1 0 1 1 0 0 1 1 ... zadání prvního operandu (10110011 = 179 dekadicky)

AND ... provede se bitový součin

0 0 1 0 0 1 1 0 ... zadání druhého operandu (00100110 = 38 dekadicky)

= [100010] ... výpočet výsledku (100010 = 34 dekadicky)

10110011

AND 00100110

= 00100010

70 Radiány Rad

Symbol **Rad**

Call up with buttons **2nd** **|**

Tlačítko **Rad** přepne výpočty goniometrických funkcí na radiány (plný úhel je $2\pi = 6.283185\dots$).

Je-li potřeba provádět výpočty nezávisle na zvolené úhlové míře, lze k převodům použít funkce **Op 72** a **Op 73**.

71 Podprogram SBR

Symbol **SBR**

Call up with the key **SBR**

Tlačítko **SBR** (Subroutine) slouží k vyvolání podprogramu. Jako parametr se zadává buď 3-místný číselný kód absolutní cílové adresy, nebo tlačítko představující návěští v programu. Je-li instrukce **SBR** použita v prováděcím režimu, podprogram se ihned spustí.

Při volání podprogramu se do zásobníku adres nejdříve uloží adresa, následující za kódem instrukce **SBR**. Poté se předá řízení podprogramu. Zásobník adres má kapacitu omezenou na 15 podprogramů.

Podprogram je ukončen instrukcí **RTN** (viz [92 Návrat z podprogramu RTN](#)). Instrukce **RTN** se vyvolá stiskem kláves **INV SBR** a zajistí návrat z podprogramu. Ze zásobníku adres se převezme původní adresa za instrukcí **SBR** a předá se na tuto adresu řízení. Pokud byl podprogram spuštěn z klávesnice, kalkulátor se zastaví.

Podprogramy lze volat též z jiného knihovního programu. V programu se uvede instrukce **Pgm** následovaná číslem programu a kódem tlačítka **A** až **F** nebo zavoláním podprogramu přes **SBR**. Použije-li se instrukce

Pgm v programu, aktivní program se nepřepne trvale (jako by tomu bylo při použití z klávesnice), přepnutí je pouze dočasné po dobu zavolání jednoho následujícího podprogramu. Po ukončení podprogramu se předá řízení zpět do původního programu.

Použitím klávesy **Ind** za kódem **SBR**, ale před uvedením adresy podprogramu, se instrukce změní na instrukci s nepřímým adresováním, **SBR Ind** (kód instrukce 26, viz [26 Podprogram s nepřímou adresou SBR Ind](#)). Blíže o způsobech adresování v kapitole [Nepřímé adresování](#).

72 Nepřímé uložení čísla do registru STO Ind

Symbol **STO Ind**

Call up with buttons **STO 2nd y^x**

Instrukcí **STO Ind** se uloží obsah displeje do datového registru stejně jako u instrukce **STO** (viz [42 Uložení čísla do datového registru STO](#)), jen namísto z parametru instrukce se číslo registru převezme z datového registru.

Instrukce se vytvoří stiskem klávesy **Ind** za klávesou **STO**, ale ještě před zadáním parametru, kterým bude 2-místné číslo registru. Blížší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

73 Indirect retrieving number from data register, RCL Ind

Symbol **RCL Ind**

Call up with buttons **RCL 2nd y^x**

Instrukcí **RCL Ind** se vyvolá číslo z datového registru stejně jako u instrukce **RCL** (viz [43 Vyvolání čísla z datového registru RCL](#)), jen namísto z parametru instrukce se číslo registru převezme z datového registru.

Instrukce se vytvoří stiskem klávesy **Ind** za klávesou **RCL**, ale ještě před zadáním parametru, kterým bude 2-místné číslo registru. Blížší informace

k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

74 Indirect add and subtract number from register, SUM Ind

Symbol **SUM** **Ind**

Call up with buttons **SUM** **2nd** **y^x**

Instrukcí **SUM** **Ind** se přičte (nebo s **INV** odečte) číslo k datovému registru stejně jako u instrukce **SUM** (viz [44 Přičtení a odečtení čísla od datového registru SUM](#)), jen namísto z parametru instrukce se číslo registru převezme z datového registru.

Instrukce se vytvoří stiskem klávesy **Ind** za klávesou **SUM**, ale ještě před zadáním parametru, kterým bude 2-místné číslo registru. Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

75 Odečtení -

Symbol **-**

Call up with the key **-**

Tlačítko **-** odečte druhý operand od prvního operandu. Jedná-li se o nejnižší úroveň výrazu, může být výpočet opakován pro jiný první operand opakovaným stiskem **=**.

76 Návěští Lbl

Symbol **Lbl**

Call up with buttons **2nd** **SBR**

Instrukcí **Lbl** lze označit místo v programu jako návěští. Za kódem **Lbl** následuje jako parametr kód tlačítka, použitého jako návěští. Lze použít jakékoliv tlačítko kromě **2nd**, číslíc **0** až **0F** a kromě **Ind**.

Na místo programu označené návěští lze skákat pomocí skokových instrukcí s uvedenou adresou, jako je **GTO**, **SBR**, **Dsz** a další. U původního kalkulátoru TI-58/59 se upřednostňovalo absolutní adresování z důvodu rychlosti, protože skok s absolutní adresou se provede ihned, zatímco vyhledání návěští v programu může nějakou dobu trvat. U kalkulátoru ET-58 se upřednostňuje používání návěští, protože jeho vyhledání je rychlé a má přednost možnost snadné relokače kódu (přemístění na jinou adresu), aniž by bylo nutné opravovat absolutní adresy skoků.

Nevyhovuje-li zadávání tlačítka jako návěští, lze kódy návěští zadávat číselně, pomocí funkce **code**. Jako parametr **code** se zadají 2 HEX číslice. Návěští může mít kód 10 až 0FE (vyjma 40 = **Ind**). Obzvláště jsou k tomu vhodné kódy 0A0 až 0FE, které nejsou kalkulátorem využity a mají význam prázdných operací. Ovšem instrukci **code** nelze použít ihned po stisku tlačítka **Lbl** či po stisku **GTO**, protože je považována za platný kód návěští. Musí se nejdříve použít náhradní návěští (např. **A**), vrátit se o krok zpět s **BST** a poté zadat kód návěští tlačítkem **code**.

Blíže k metodám adresování viz kapitola [Nepřímé adresování](#).

77 Větší nebo rovno $x \geq t$

Symbol **$x \geq t$**

Call up with buttons **2nd** **4**

Instrukce **$x \geq t$** umožňuje porovnat registr X (obsah displeje) s pomocným registrem T (nastaveným tlačítkem **$x \leftrightarrow t$**). Je-li registr X větší nebo roven registru T, provede se skok na adresu, která je zadána jako parametr instrukce. Adresou může být absolutní adresa nebo návěští.

Instrukce **$x \geq t$** nemá zvláštní kód pro nepřímé adresování. Přesto nepřímé adresování umožňuje a to tak, že kód **Ind** se uloží do programu za kód **$x \geq t$** . Bude-li podmínka splněna, načte se ze zadaného registru absolutní adresa nebo kód návěští, tak jak je podrobněji popsáno v kapitole [Nepřímé adresování](#).

Je-li před kódem **$x \geq t$** zadán prefix **INV**, provede se inverzní funkce - skok

na zadanou adresu se provede v případě, že registr X je menší než registr T.

Další informace viz instrukce [67 Rovno x=t](#). Doporučení k testu rovnosti platí i pro instrukci **x>=t**.

78 Statistika Stat

Symbol **Stat**

Call up with buttons **2nd** **5**

Instrukce **Stat** slouží k zadávání dat při provádění statistických výpočtů (průměr, korelace, variace) a při výpočtu lineární regrese (proložení aproximační přímkou). Instrukce používá datové registry R01 až R06 k ukládání mezivýpočtů. Před použitím je nutné registry nejdříve vynulovat příkazem **INV** **CMs**, který vynuluje R01 až R06, spolu s X a T. Jinou alternativou je podprogram **CLR** v knihovním programu ML-01 (vyvolá se pomocí **Pgm** **01** **SBR** **CLR**).

Při vkládání statistických dat po dvojicích (x, y) se nejdříve zapíše hodnota 'x' a stiskem **x<>t** se přenese do registru T. Poté se zapíše hodnota 'y' a stiskem **Stat** se obě hodnoty x a y uloží. Na displeji (v registru X) se objeví počet dosud vložených položek 'n'. Obsah registru T (hodnota x) se instrukcí **Stat** zvýší o 1. To z důvodu, že pokud se hodnoty x mají lišit o 1, není potřeba je vkládat, stačí vložit počáteční hodnotu x do registru T a poté zapisovat už jen hodnoty y. Nejsou-li potřeba vyhodnocovat dvojice hodnot (x, y), postačí zadávat pouze hodnotu y.

Uvede-li se před instrukcí **Stat** prefix **INV**, vložená hodnota se naopak odečte. Tak lze opravit chybně vloženou hodnotu - vloží se hodnota x chybného údaje, stiskne se **x<>t**, vloží se hodnota y chybného údaje a stiskem **INV** **Stat** se chybné údaje odečtou. Poté lze pokračovat novým správným údajem. Nejsou-li potřeba vyhodnocovat dvojice hodnot (x, y), postačí zadávat pouze hodnotu y chybného údaje.

Použité registry:

R01 suma y

R04 suma x

R02 suma y^2

R05 suma x^2

R03 počet položek n

R06 suma $x \cdot y$

79 Průměr Mean

Symbol **Mean**

Call up with buttons **2nd** **6**

Instrukce **Mean** vypočte průměr z hodnot 'x' a 'y' zadaných pomocí statistické funkce **Stat** (podrobnosti viz [78 Statistika Stat](#)). Na displeji se zobrazí průměr z hodnot 'y'. Po stisku **x<>t** se z registru T zobrazí průměr z hodnot 'x'.

Uvedením prefixu **INV** před funkcí **Mean** se vypočte směrodatná odchylka. Na displeji se zobrazí směrodatná odchylka hodnot 'y' **devy** = $\sqrt{(\text{sum}(y^2) - \text{sum}(y)^2/N)/(N-1))}$, v registru T je směrodatná odchylka hodnot 'x' **devx** = $\sqrt{(\text{sum}(x^2) - \text{sum}(x)^2/N)/(N-1))}$.

Example:

INV **CMs** [0] ... vynulování registrů

9 **6** **Stat** [1] ... 1. údaj 96

8 **1** **Stat** [2] ... 2. údaj 81

9 **7** **Stat** [3] ... 3. údaj je chybný

9 **7** **INV** **Stat** [2] ... zrušení 3. údaje

8 **7** **Stat** [3] ... 3. opravný údaj 87

7 **0** **Stat** [4] ... 4. údaj 70

9 **3** **Stat** [5] ... 5. údaj 93

7 **7** **Stat** [6] ... 6. údaj 77

Mean [82] ... průměr zadaných hodnot = 82

INV Mean [9.87927...] ... směrodatná odchylka = 9.87927...

Op 11 [81.333...] ... variace = 81.333...

RCL 01 [504] ... součet všech hodnot = 504

7A Podmíněný skok IF

Symbol **IF**

Call up with buttons **2nd 2nd SBR**

Instrukce **IF** je skok podmíněný porovnáním registrů. Na rozdíl od instrukcí **x=t** a **x>=t** se zde nepracuje s číslem na displeji, ale porovnávají se registry mezi sebou. Jako operandy je možné použít datové registry nebo HIR registry, v přímém i nepřímém adresování.

Prvním parametrem, následujícím za kódem **IF**, je dvoumístné HEX číslo. První (vyšší) číslice představuje kód prováděné operace. Druhá (nižší) číslice představuje index prvního operandu. Prvním operandem může být datový registr R00 až R15, HIR registr H0 až H15, a to buď jako přímé registry nebo nepřímé. Jedná-li se o nepřímé adresování s HIR registrem, je z HIR registru načten index registru, ale jako indexovaný registr je použit datový registr (ne HIR registr). HIR registr je použit jako ukazatel do hlavní uživatelské paměti datových registrů.

Druhým parametrem je dekadické číslo, představující číslo registru nebo dekadickou konstantu. Registrem může být datový registr R00 až R99 nebo HIR registr H0 až H15. Jedná-li se o nepřímé adresování, je adresován vždy datový registr, bez ohledu na to, že index může být načten z HIR registru.

Třetím parametrem instrukce je cílová adresa, na kterou program skočí při splnění podmínky. Adresa může být absolutní, návěští nebo nepřímá adresa. Nepřímá v případě, že jako třetí parametr je uveden kód **Ind** následovaný číslem registru. Nepřímou adresu lze použít nezávisle na tom, zda je použita přímá instrukce **IF** nebo nepřímá instrukce **IF Ind**.

Pokud se před instrukcí **IF** stiskne tlačítko prefixu **INV**, instrukce **IF** se

změní na instrukci **IF Ind** s nepřímým adresováním (podrobnosti viz [6D Nepřímá podmínka IF Ind](#)).

*Pozor, instrukce IF umožňuje zadat parametry v HEX kódu a vložit tak do kódu bajt s hodnotou 0FF. Je-li to možné, vyhýbejte se takové hodnotě bajtu, protože by byl interpretován jako prázdné místo a poškodil by se během posunu programu v paměti s **Ins** či **Del**.*

Kódy operací IF, podle 1. číslice 1. parametru

Kód operace se skládá ze 4 bitů: bit 0: menší <, bit 1: rovno =, bit 2: větší >, bit 3: HIR registr. Pro kódy 0 až 7 se pracuje s datovými registry. Kódy 8 až 0F dělají stejnou operaci jako kódy 0 až 7, ale používají HIR registry. To platí pro oba registry porovnání. Nelze porovnávat datový registr (v přímé adresaci) s HIR registrem. V následující tabulce je uveden kód pro datový registr a v závorce kód stejné operace pro HIR registr. Při porovnání s konstantou (kódy 0, 7, 8 a 0F) je dekadická konstanta v rozsahu 00 až 99 (příp. až 0F9) uvedena jako druhý parametr instrukce. U nepřímé instrukce **IF Ind** se hodnota konstanty nenačítá nepřímo z registru, ale bere se také přímo z druhého parametru.

0 (8) <= konstanta

1 (9) <

2 (A) =

3 (B) <=

4 (C) >

5 (D) <> (není rovno)

6 (E) >=

7 (F) > konstanta

Example - vyhledání hodnoty >8 v datových registrech

CMs 9 STO 13 ... vynulování registrů, do registru R13 uloží číslo 9

3 0 HIR 02 ... (30 STO H2) příprava počtu registrů (30) do H2

RST LRN ... aktivace režimu programování

CLR HIR 01 ... (0 STO H1) příprava počátečního indexu 0 do H1

Lbl **Inx** ... návěští začátku cyklu

IF **Ind** **0F1** **08** **=** ... pokud nepřímý obsah $H1^* > 8$, skok na =

... Note: **IF** **Ind** se zapíše posloupností **INV** **IF**

HIR **71** ... (Inc H1) inkrementace indexu v H1

IF **91** **02** **Inx** ... pokud $H1 < H2$, skok na Inx

CLR **1/x** **R/S** ... indikace chyby a stop (nenalezeno)

Lbl **=** ... návěští pro případ úspěchu

HIR **11** **R/S** ... zobrazí index nalezeného registru a stop

LRN ... deaktivace režimu programování

RST **R/S** [13] ... TEST: reset a start programu, nalezeno v R13

8 **STO** **13** ... do R13 uloží hodnotu která by neměla projít

RST **R/S** [9.999+9999] ... TEST: tentokrát nenalezeno, bliká chyba

7E Bitový exkluzivní součet XOR

Symbol **XOR**

Call up with buttons **2nd** **2nd** **=**

Instrukce **XOR** provede bitovou operaci XOR (exkluzivní součet) mezi prvním a druhým operandem. Jedná-li se o nejnižší úroveň výrazu, může být výpočet opakován pro jiný první operand opakovaným stiskem **=**.

Bitový exkluzivní součet znamená, že výsledkem operace je bit '1' pouze v případě, když se oba vstupní bity liší.

Example:

BIN ... přepnutí na bitový mód zobrazení

1 **0** **1** **1** **0** **0** **1** **1** ... zadání prvního operandu ($10110011 = 179$ dekadicky)

XOR ... provede se bitový exkluzivní součet

00100110 ... zadání druhého operandu (00100110 = 38 dekadicky)

= [10010101] ... výpočet výsledku (10010101 = 149 dekadicky)

10110011
XOR 00100110
= 10010101

80 Grady Grad

Symbol **Grad**

Call up with buttons **2nd** **+**

Tlačítko **Grad** přepne výpočty goniometrických funkcí na grady (plný úhel je 400).

Je-li potřeba provádět výpočty nezávisle na zvolené úhlové míře, lze k převodům použít funkce **Op 72** a **Op 73**.

81 Reset RST

Symbol **RST**

Call up with the key **RST**

Tlačítko **RST** slouží k resetování ukazatele programu, tj. nastavení ukazatele na 0. Je-li vybrán program z knihovního modulu, deaktivuje se a přepne se zpět na uživatelský hlavní program. Běží-li program z knihovního modulu, běh programu se zastaví. Běží-li hlavní program, pokračuje v činnosti od adresy 0.

Instrukce **RST** nuluje stavy přepínačů, kromě přepínače 15, který indikuje chybu E.

82 Interní instrukce HIR

Symbol **HIR**

Call up with buttons **2nd** **INV**

U původního kalkulátoru TI-58/59 byla **HIR** instrukce skrytou instrukcí, nedokumentovanou v manuálech. Ke své činnosti využívala registry zásobníku aritmetických operací.

U kalkulátoru ET-58 se **HIR** instrukce stala jednou z hlavních instrukcí programů knihovny. Používá samostatnou sadu 16 řídicích registrů H0 až H15, které (na rozdíl od původního TI-58/59) nejsou sdílené se zásobníkem aritmetických operací, jsou to zcela samostatné a nezávislé registry. Jsou určeny především jako pracovní řídicí registry, zatímco datové registry R00 až R99 zůstávají plně k dispozici pro uživatelská data. Množina funkcí **HIR** instrukce byla podstatně rozšířena a zahrnuje podobné vybavení jako instrukce pro práci s datovými registry.

Za kódem **HIR** instrukce následuje jako parametr bajt v HEX kódu. První HEX číslice (vyšší) představuje kód instrukce, druhá HEX číslice (nižší) je číslo HIR registru H0 až H15, se kterým se má provést operace.

Uvedením kódu **Ind** za instrukcí **HIR**, ale ještě před zadáním parametru, se instrukce **HIR** změní na nepřímou instrukci **HIR Ind** (s kódem 27, viz [27 Nepřímá interní instrukce HIR Ind](#)). Nepřímá instrukce pracuje s datovým registrem R00 až R99, jehož index je obsažen v HIR registru H0 až H15 daným druhou číslicí parametru **HIR** instrukce. Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

Kódy operací HIR instrukcí, podle 1. číslice parametru:

- 0** ... STO, uloží obsah displeje do HIR registru
- 1** ... RCL, vyvolá obsah HIR registru
- 2** ... round, zaokrouhlí HIR registr na nejbližší celé číslo
- 3** ... SUM, přičte displej k HIR registru
- 4** ... Prd, vynásobí HIR registr displejem
- 5** ... INV SUM, odečte displej od HIR registru

- 6 ... INV Prd**, vydělí HIR registr displejem
- 7 ... Inc**, inkrementuje (zvýší o 1) HIR registr
- 8 ... Dec**, dekrementuje (sníží o 1) HIR registr
- 9 ... Exc**, zamění displej s HIR registrem
- A ... GTO**, skok na adresu z HIR registru (absolutní nebo návěští)
- B ... SBR**, vyvolání podprogramu podle HIR registru
- C a ... $x \leq 0$** , skok na uvedenou adresu, je-li HIR registr ≤ 0
- D a ... $x > 0$** , skok na uvedenou adresu, je-li HIR registr > 0
- E a ... DJNZ** (Decrement and Jump if Not Zero, význam jako DSZ), dekrementace/inkrementace HIR registru a skok na uvedenou adresu, pokud HIR registr není 0.
- F a ... DJZ** (Decrement and Jump if Zero, význam jako INV DSZ), dekrementace/inkrementace HIR registru a skok na uvedenou adresu, pokud HIR registr je 0.

Instrukce **HIR** **20** se u původního TI-58/59 používala k větvení programu interního mikrokódu. Umožňovala, podle přednastaveného interního registru, provést buď relativní skok nebo ukončení funkce. U ET-58 pozbývá význam a je proto využita k jiným účelům - zaokrouhlení HIR registru. Zaokrouhlení je užitečné v případě mnoha opakovaných výpočtů, kdy se chyba zaokrouhlení může naakumulovat a může selhat porovnání výsledku operace na shodu. Typickým příkladem jsou cykly, kdy se opakovaně přičítá či odečítá konstanta od registru. Akumulací chyby nepřesnosti se výsledná hodnota může odchýlit od testovaného počtu cyklů. Průběžným zaokrouhlováním výsledku na celá čísla se bude chyba korigovat.

Funkce A (GTO) a B (SBR) používají jako cílovou adresu obsah HIR registru. Adresa může být absolutní, tj. dekadické číslo v rozsahu 0 až 999, nebo kód návěští. Jedná-li se o kód návěští, použije se dekadická hodnota kódu tlačítka návěští, vynásobená číslem 256. Bližší popis naleznete v kapitole [Nepřímé adresování](#).

Funkce C ($x \leq 0$) až F (DJZ) mají navíc uveden ještě druhý parametr, adresu skoku. Adresa může být absolutní nebo návěští. Doplněním kódu **Ind** může být adresa nepřímá, obsažená v uvedeném datovém registru.

Cykly E (DJNZ) a F (DJZ) mají podobnou funkci jako instrukce **DSZ** a **INV**

DSZ (viz [97 Programová smyčka Dsz](#)), ale na rozdíl od nich používají HIR registr. Je-li obsah registru před operací větší než 0, hodnota registru se sníží o 1. Je-li menší než 0, hodnota se zvýší o 1. Byl-li obsah registru 0, hodnota se nezmění. Pokud je výsledkem operace nula nebo operace překročila hranici nuly, provede se operace pro nulu podle zvolené funkce. Na závěr se hodnota registru zaokrouhlí na celé číslo, což zabrání akumulaci chyby při opakování operací.

*Pozor, instrukce HIR umožňuje zadat parametr v HEX kódu a vložit tak do kódu bajt s hodnotou 0FF (operace DJZ H15). Je-li to možné, vyhýbejte se takové hodnotě bajtu, protože by byl interpretován jako prázdné místo a poškodil by se během posunu programu v paměti s **Ins** či **Del**.*

Example - naplnění registrů R00 až R99 čísly 100 až 199

RST **LRN** ... aktivace programovacího režimu
Lbl **A** ... návěští podprogramu
0 **HIR** **01** ... (STO H1) počáteční index registrů = 0
1 **0** **0** ... ukládaná hodnota (a současně počet registrů)
HIR **02** ... (STO H2) čítač smyčky = 100
Lbl **=** ... návěští začátku smyčky
(**HIR** **Ind** **01** ... (STO Ind H1) zápis do registru adresovaného H1
+ **1** **)** ... inkrementace čísla na displeji
HIR **71** ... (Inc H1) inkrementace indexu H1
HIR **E2** **=** ... (DJNZ H2 =) dekrementace H2 a skok na = když není 0
RTN ... konec podprogramu (**RTN** se zapíše jako **INV** **SBR**)
LRN ... ukončení programovacího režimu
A [200] ... spuštění programu
RCL **99** [199] ... kontrola registru R99 zda obsahuje číslo 199
RCL **13** [113] ... kontrola registru R13 zda obsahuje číslo 113

83 Nepřímý skok GTO Ind

Symbol **GTO** **Ind**

Call up with buttons **GTO** **2nd** **y^x**

Instrukce **GTO** **Ind** provádí skok v programu stejně jako instrukce **GTO** (viz [61 Skok GTO](#)), jen namísto z parametru instrukce se adresa skoku převezme z datového registru. Registr může obsahovat jak absolutní adresu (000 až 999), tak i návěští (dekadický kód návěští * 256).

Instrukce se vytvoří stiskem klávesy **Ind** za klávesou **GTO**, ale ještě před zadáním parametru, kterým bude 2-místné číslo registru. Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

Example

RST **LRN** ... aktivace režimu programování

GTO **Ind** **01** ... nepřímý skok na adresu z registru R01

1 **R/S** ... adresa 2, zobrazí číslo 1

2 **R/S** ... adresa 4, zobrazí číslo 2

3 **R/S** ... adresa 6, zobrazí číslo 3

LRN .. deaktivace režimu programování

2 **STO** **01** **RST** **R/S** [1] ... test, skočí na adresu 2 a zobrazí 1

CLR **4** **STO** **01** **RST** **R/S** [2] ... test, skočí na adresu 4 a zobrazí 2

CLR **6** **STO** **01** **RST** **R/S** [3] ... test, skočí na adresu 6 a zobrazí 3

84 Nepřímá speciální operace Op Ind

Symbol **Op** **Ind**

Call up with buttons **2nd** **Op** **2nd** **y^x**

Instrukce **Op** **Ind** provede speciální operaci stejně jako instrukce **Op** (viz kapitola [69 Speciální operace Op](#) a kapitola [Speciální operace Op](#)), jen namísto z parametru instrukce se kód operace převezme z datového

registru.

Instrukce se vytvoří stiskem klávesy **Ind** za klávesou **Op**, ale ještě před zadáním parametru, kterým bude 2-místné číslo registru. Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

85 Přičtení +

Symbol **+**

Call up with the key **+**

Tlačítko **+** přičte druhý operand k prvnímu operandu. Jedná-li se o nejnižší úroveň výrazu, může být výpočet opakován pro jiný první operand opakováním stiskem **=**.

86 Nastavení a nulování přepínače StFlg

Symbol **StFlg**

Call up with buttons **2nd** **RST**

Instrukce **StFlg** (Set Flag) zapne přepínač 0 až 15, uvedený jako parametr instrukce. Parametrem je číslo 00 až 0F. Použitím prefixu **INV** před instrukcí **StFlg** se provede opačná operace - vypnutí přepínače.

Instrukce **StFlg** nemá kód pro nepřímé adresování. Nepřímé adresování lze dosáhnout tak, že se namísto čísla registru 00..0F uvede kód **Ind** následovaný číslem registru, který obsahuje číslo přepínače.

Instrukce **RST** nuluje stavy přepínačů, kromě přepínače 15, který indikuje chybu E.

Některé přepínače mají speciální význam:

7 ... přepínač lze nastavit operacemi **Op** **18** a **Op** **19** podle stavu chyby

8 ... je-li přepínač 8 zapnutý, program se zastaví při vzniku měkké chyby E.

Není-li zapnutý, kalkulátor sice indikuje chybu, ale pokračuje ve výpočtu. Při vzniku tvrdé chyby F se program zastaví vždy.

15 ... přepínač 15 je přímo propojen s indikací chyby E. Stav chyby lze přepínačem 15 testovat, zapínat i vypínat.

Note: Po resetování indikace chyby nulováním přepínače 15 může na displeji zůstat svítit znak E. Nejde o závadu, po první změně obsahu displeje znak zmizí.

87 Test přepínače IfFlg

Symbol **IfFlg**

Call up with buttons **2nd** **1**

Instrukce **IfFlg** (If Flag) provede skok na adresu danou třetím parametrem v případě, že přepínač 0 až 15, zadaný druhým parametrem, je zapnutý. Uvedením prefixu **INV** před instrukcí **IfFlg** se provede opačná funkce - skok se provede v případě vypnutého přepínače.

Instrukce **IfFlg** nemá kód pro nepřímé adresování. Nepřímé adresování čísla přepínače lze dosáhnout tak, že se namísto druhého parametru uvede kód **Ind** následovaný číslem registru, který bude obsahovat číslo přepínače. Podobně lze dosáhnout nepřímé adresování skoku - namísto třetího parametru se uvede kód **Ind** následovaný číslem registru, který obsahuje cílovou adresu nebo návěští. Blíže o nepřímém adresování viz kapitola [Nepřímé adresování](#).

Instrukce **RST** nuluje stavy přepínačů, kromě přepínače 15, který indikuje chybu E.

Example - nepřímé zobrazení stavu přepínače:

RST **LRN** ... aktivace módu programování

Lbi **IfFlg** ... návěští začátku podprogramu

CLR ... přednastaví výsledek 0 (vypnutý přepínač)

INV IfFlg Ind 01 ... není-li přepínač s indexem R01, skok na **=**

1 ... výsledek bude 1

x<>t x<>t ... malá korekce pro ukončení módu editace

Lbl ... sem skočí v případě vypnutého přepínače

RTN ... konec podprogramu (**INV SBR**)

Lbl A ... návěští testu přepínače 1

1 STO 01 ... do R01 uloží číslo přepínače 1

SBR IfFlg ... test stavu přepínače 1

RTN ... konec podprogramu (**INV SBR**)

Lbl B ... návěští testu přepínače 2

2 STO 01 ... do R01 uloží číslo přepínače 2

SBR IfFlg ... test stavu přepínače 2

RTN ... konec podprogramu (**INV SBR**)

LRN ... ukončení módu programování

StFlg 2 ... nastavení přepínače 2

A [0] ... test přepínače 1, je vypnutý

B [1] ... test přepínače 2, je zapnutý

88 Převody minut a sekund DMS

Symbol **DMS**

Call up with buttons **2nd 2**

Instrukcí **DMS** lze převést čas nebo úhel vyjádřený pomocí minut a sekund (vteřin) na desetinné číslo. Vstupem funkce je desetinné číslo DD.MMSS, mající na pozici celých čísel celý počet hodin či stupně, na prvních dvou desetinných místech je počet minut a na dalších dvou desetinných místech je počet sekund (vteřin). Desetinná místa sekund lze doplnit jako další desetinné číslice. Výstupem funkce je desetinné číslo představující počet

hodin či stupně DD.DDDD, vyjádřených desetinným číslem.

Uvedením prefixu **INV** před instrukcí **DMS** se provede opačná operace - čas nebo úhel vyjádřený pomocí desetinného čísla se převede na údaj minut a sekund (vteřin). Vstupem funkce je desetinné číslo představující počet hodin či stupně DD.DDDD. Výstupem je číslo DD.MMSS, mající na pozici celých čísel celý počet hodin či stupně, na prvních dvou desetinných místech je počet minut a na dalších dvou desetinných místech je počet sekund (vteřin). Není-li výsledkem celý počet sekund, jsou desetinná místa sekund doplněna jako další desetinné číslice.

Example, součet času:

12**.****30****.****23** ... čas je 12:30:23 (12 hodin, 30 minut a 23 sekund)

DMS [12.50638...] ... převod na hodiny vyjádřené v desetínách

+ **3****.****45****.****12** **DMS** [3.7533...] ... plus 3 hodiny, 45 minut a 12 sekund

= **INV** **DMS** [16.1535] ... výsledný čas 16 hodin, 15 minut a 35 sekund

89 Ludolfovo číslo pi

Symbol **pi**

Call up with buttons **2nd** **3**

Tlačítko **pi** slouží k zadání konstanty "Ludolfovo číslo pi", které má hodnotu 3.141592653589793238.

8A Operace s registry Reg

Symbol **Reg**

Call up with buttons **2nd** **2nd** **RST**

Instrukce **Reg** umožňuje přímou manipulaci s registry, bez nutnosti používat obsah displeje (registr X).

Za instrukcí **Reg** následují dva parametry. První parametr je v HEX tvaru. Jeho první (vyšší) číslice představuje kód operace 0 až 0F. Druhá (nižší) číslice udává cílový operand operace. Cílovým operandem je datový registr R00 až R15 nebo HIR registr H0 až H15. Cílový operand může být též nepřímý. Druhý parametr udává zdrojový operand. Může jím být datový registr R00 až R99 nebo HIR registr H0 až H15.

Uvedením prefixu **INV** před instrukcí **Reg** se provede inverzní nebo alternativní operace.

Uvedením kódu **Ind** za prvním parametrem instrukce **Reg**, ale ještě před zadáním druhého parametru, se instrukce změní na nepřímou instrukci **Reg Ind** (číselný kód 6C, viz kapitola [6C Nepřímá operace s registry Reg Ind](#)). Nepřímá instrukce **Reg Ind** umožňuje nepřímé adresování druhého parametru operace (zdrojový registr). Nepřímé adresování neovlivní způsob adresování prvního (cílového) operandu, ten je vždy adresován typem instrukce. A také neovlivní konstantu, ta se vždy načítá z parametru instrukce. Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

Kódy operací:

Kód operace je určen první (vyšší) číslicí prvního parametru. Bit 0 a bit 1 určuje číslo operace. Bit 2 označuje nepřímé adresování prvního (cílového) operandu. Bit 3 označuje HIR registry. V případě HIR registrů budou jako oba operandy použity HIR registry, nelze provést přímou operaci mezi datovým a HIR registrem. Je-li některý z operandů adresován nepřímo, může být index uložen buď v datovém nebo HIR registru, ovšem adresovaný registr je vždy datovým registrem. HIR registr je použit jako ukazatel do datových registrů.

V následující tabulce představuje kód operace 0 až 3 přímý datový registr, kód 4 až 7 je nepřímý datový registr, kód 8 až 0B je přímý HIR registr, kód 0C až 0F je nepřímý HIR registr (tj. HIR registr je ukazatel do datových registrů).

0, 4, 8, 0C ... MOV kopie obsahu, s INV je Exc záměna registrů

1, 5, 9, 0D ... SUM přičtení, s INV je odečtení

2, 6, 0A, 0E ... Prd vynásobení, s INV vydělení

3, 7, 0B, 0F ... konstanta z 2. parametru BCD, s INV je negativní konstanta

V případě nastavení konstanty je konstanta převzata z 2. parametru instrukce, jako dekadické BCD číslo (např. bajt 99 znamená hodnotu 99). Význam není ovlivněn nepřímým adresováním, i v tom případě se konstanta použije přímo z 2. parametru. Při použití prefixu **INV** se konstanta uloží jako negativní číslo.

*Note: Instrukce **Reg** umožňuje zadat parametr s hodnotou OFF (operace OFF = načtení konstanty do HIR registru H15). Pokud možno, vyhněte se používání bajtu OFF. Při přesunech paměti s **Ins** nebo **Del** by byl parametr OFF považován za prázdné místo a kód by byl poškozen.*

Example:

1 **0** **HIR** **01** ... (STO H1) do HIR registru H1 uloží číslo 10

Reg **B2** **13** ... do HIR registru H2 uloží konstantu 13

Reg **A1** **02** ... (Prd H1 H2) HIR registr H1 vynásobí HIR registrem H2

HIR **11** [130] ... (RCL H1) test, obsah HIR registru H1 je 130

8B Hexadecimální mód HEX

Symbol **HEX**

Call up with buttons **2nd** **2nd** **1**

Instrukce **HEX** přepne mód displeje do hexadecimálního zobrazení. Mantisa čísla se zobrazí i zadává v HEX kódu, číslice 0 až 0F, a to včetně desetinných míst mantisy. Exponent je vždy dekadické číslo.

Uvedením prefixu **INV** před instrukcí **HEX** se aktivuje pomocný ladicí mód, kdy se v prvním řádku displeje zobrazí mantisa zobrazeného čísla v interním formátu, jako 16 hexadecimálních číslic. Tímto způsobem lze zobrazit i skrytá desetinná místa mantisy. První číslice zleva je nejvyšší číslice. Nejvýznamnější bit mantisy s hodnotou '1' je skrytý a je nahrazen znaménkovým bitem. Ladicí mód se vypne volbou módu zobrazení bez prefixu **INV**.

V technickém módu Eng je exponent násobkem čísla 4.

8C Binární mód BIN

Symbol **BIN**

Call up with buttons **2nd** **2nd** **2**

Instrukce **BIN** přepne mód displeje do binárního zobrazení. Mantisa čísla se zobrazí i zadává v BIN kódu, číslice 0 až 1, a to včetně desetinných míst mantisy. Exponent je vždy dekadické číslo.

Uvedením prefixu **INV** před instrukcí **BIN** se aktivuje pomocný ladicí mód, kdy se v prvním řádku displeje zobrazí mantisa zobrazeného čísla v interním formátu, jako 16 hexadecimálních číslic. Tímto způsobem lze zobrazit i skrytá desetinná místa mantisy. První číslice zleva je nejvyšší číslice. Nejvýznamnější bit mantisy s hodnotou '1' je skrytý a je nahrazen znaménkovým bitem. Ladicí mód se vypne volbou módu zobrazení bez prefixu **INV**.

V technickém módu Eng je exponent násobkem čísla 4.

8D Oktalový mód OCT

Symbol **OCT**

Call up with buttons **2nd** **2nd** **3**

Instrukce **OCT** přepne mód displeje do oktalového zobrazení. Mantisa čísla se zobrazí i zadává v OCT kódu, číslice 0 až 7, a to včetně desetinných míst mantisy. Exponent je vždy dekadické číslo.

Uvedením prefixu **INV** před instrukcí **OCT** se aktivuje pomocný ladicí mód, kdy se v prvním řádku displeje zobrazí mantisa zobrazeného čísla v interním formátu, jako 16 hexadecimálních číslic. Tímto způsobem lze zobrazit i skrytá desetinná místa mantisy. První číslice zleva je nejvyšší číslice. Nejvýznamnější bit mantisy s hodnotou '1' je skrytý a je nahrazen znaménkovým bitem. Ladicí mód se vypne volbou módu zobrazení bez prefixu **INV**.

V technickém módu Eng je exponent násobkem čísla 3.

8E Bitový součet OR

Symbol **OR**

Call up with buttons **2nd** **2nd** **+**

Instrukce **OR** provede bitovou operaci OR (bitový součet) mezi prvním a druhým operandem. Jedná-li se o nejnižší úroveň výrazu, může být výpočet opakován pro jiný první operand opakovaným stiskem **=**.

Bitový součet znamená, že výsledkem operace je bit '1' v případě, kdy má alespoň jeden vstupní bit hodnotu '1'.

Example:

BIN ... přepnutí na bitový mód zobrazení

1**0****1****1****0****0****1****1** ... zadání prvního operandu (10110011 = 179 dekadicky)

OR ... provede se bitový součet

0**0****1****0****0****1****1****0** ... zadání druhého operandu (00100110 = 38 dekadicky)

= [10110111] ... výpočet výsledku (10110111 = 183 dekadicky)

	10110011
OR	00100110
=	10110111

91 Start a stop programu R/S

Symbol **R/S**

Call up with the key **R/S**

Tlačítkem **R/S** lze spustit nebo zastavit probíhající program. Při spuštění se program začne provádět od aktuálně nastaveného ukazatele programu (aktuální adresu lze zjistit přepnutím do programovacího módu **LRN**).

Program po zastavení nemusí být vždy schopen pokračovat v běhu - může dojít ke ztrátě návratové adresy z podprogramu nebo může zůstat přepnutý jiný program knihovny.

92 Návrat z podprogramu RTN

Symbol **RTN**

Call up with buttons **INV** **SBR**

Instrukce **RTN** slouží k návratu programu z podprogramu. Ze zásobníku adres se převezme původní adresa za instrukcí, která podprogram vyvolala (instrukce **SBR** nebo **A** až **F**), a předá se na tuto adresu řízení. Pokud byl podprogram spuštěn z klávesnice, kalkulátor se zastaví.

Instrukce **RTN** se z klávesnice zapíše stiskem **INV** **SBR** ("inverze podprogramu").

93 Desetinná tečka .


Symbol **.**


Call up with the key **.**


Tlačítko **.** je oddělovač celočíselných číslic mantisy a desetinných číslic mantisy. Je-li tlačítko **.** stisknuto během editace exponentu čísla, přejde editace zpět k zadávání mantisy čísla.

Je-li před tečkou **.** uveden prefix **INV**, provede se jen zahájení editace čísla podobně jako u posloupnosti **EE INV EE**, výhodou ovšem je, že není měněn mód exponentu. Tímto postupem lze snadno odstranit skryté číslice - zaokrouhlit číslo. Jinou alternativou je operace **Op 82**, která neponechá číslo v editovatelném tvaru.

94 Změna znaménka +/-


Symbol 


Call up with the key 



Tlačítko  změni znaménko čísla na displeji. Je-li stisknuto během zadávání exponentu čísla, změni znaménko exponentu.


95 Provedení výpočtu =

Symbol 





Call up with the key 



Tlačítko  slouží k uzavření otevřených aritmetických operací a provedení výpočtu.





Opakovaným stiskem tlačítka  lze opakovat poslední prováděnou operaci nejnižší úrovně. Prvním operandem je číslo na displeji, druhým operandem je číslo zadané během operace jako druhý operand (nebo druhý výsledek mezivýpočtu). Operandy lze zaměnit tlačítkem .

Upozornění: Některé programy původní TI-58/59 s opakováním operací nepočítají, použijí klávesu  vícekrát a tím může vzniknout chybný výpočet. Při importu programů může být nutné tuto skutečnost zkontrolovat a ošetřit.



Example:

    [30] ... $5 \times 6 = 30$

  [42] ... $7 \times 6 = 42$

    [3] ... $9 / 3 = 3$

   [4] ... $12 / 3 = 4$

  [3] ... záměna operandů, druhým operandem nyní bude číslo 2

   [8] ... $16 / 2 = 8$

97 Programová smyčka Dsz

Symbol **Dsz**

Call up with buttons **2nd** **0**

Instrukce **Dsz** (Decrement and Skip if Zero) slouží k opakovanému provádění programu pomocí smyčky s čítáním průchodů v registru. Za instrukcí **Dsz** následují 2 parametry. Prvním parametrem je číslo datového registru 0 až 0F (odpovídá registru R00 až R15). Druhým parametrem je adresa skoku - buď jako absolutní adresa nebo jako návěští.

Zjednodušeně funkce **Dsz** spočívá v tom, že dekrementuje (sníží o 1) uvedený registr a pokud dosáhl nuly, přeskočí adresu uvedenou jako druhý parametr a pokračuje v činnosti. Zřejmější význam může být ve zkratce DJNZ (Decrement and Jump if Not Zero) - dekrementuj o 1 a pokud není 0, skoč (tedy opakování smyčky, pokud registr není 0).

Uvede-li se před instrukcí **Dsz** prefix **INV**, provede se opačný význam instrukce - adresa se přeskočí v případě, že výsledek dekrementace není nula. Nebo jinak DJZ (Decrement and Jump if Zero) - dekrementuj o 1 a pokud je 0, skoč (tedy skok jinam při ukončení smyčky).

Instrukce **Dsz** nemá zvláštní kód pro nepřímé adresování. Nepřímé adresování lze dosáhnout uvedením kódu **Ind** před prvním parametrem. V tom případě se index čítacího registru převezme z registru uvedeného jako parametr kódu **Ind**, a/nebo uvedením kódu **Ind** před druhým parametrem, v tom případě se adresa skoku převezme z registru uvedeného jako parametr druhého kódu **Ind**. Viz též [Nepřímé adresování](#).

Alternativou instrukce **Dsz** je **HIR** instrukce s kódy 0E0 až 0EF (DJNZ) a s kódy 0F0 až 0FF (DJZ), které namísto datových registrů používají HIR registry. Viz [82 Interní instrukce HIR](#).

Instrukce **Dsz** smyčky funguje přesněji následujícím způsobem. Je-li obsah registru před operací větší než 0, hodnota registru se sníží o 1. Je-li menší než 0, hodnota se zvýší o 1. Byl-li obsah registru 0, hodnota se nezmění. Pokud je výsledkem operace nula nebo operace překročila hranici nuly, provede se operace pro nulu podle zvolené funkce. Což znamená, že pokud nedosáhl výsledek dekrementace/inkrementace nulu, smyčka se opakuje. S prefixem **INV** se provede skok naopak pokud

výsledek operace dosáhl nulu (nebo ji přesáhl).

Na závěr se hodnota registru zaokrouhlí na celé číslo, což zabrání akumulaci chyby při opakování operací. Tím se liší instrukce **DSZ** od funkce původního kalkulátoru TI-58/59. Zaokrouhlení je vyžádáno tím, že kalkulátor ET-58 pracuje s binárními čísly. Původní TI-58/59 pracuje s BCD čísly, a tak u něj probíhá automatické zaokrouhlování na čísla vyjádřená dekadickými číslicemi. Tato odchylka funkčnosti se sice v praxi zpravidla nijak neprojeví, ale je možné že některý program může očekávat, že se desetinná čísla smyčky nezaokrouhlují.

Example - vymazání všech registrů:

RST **LRN** ... aktivace programovacího módu

9 **9** **STO** **00** ... čítač do registru R00, index posledního registru

CLR ... příprava čísla 0 k uložení do registrů

Lbl **CLR** ... návěští začátku smyčky

STO **Ind** **00** ... do registru s indexem z R00 se uloží 0 z displeje

DSZ **0** **CLR** ... dekrementuj R00 a není-li nula, opakuj smyčku

R/S ... zastavení programu

LRN ... ukončení programovacího módu

9 **STO** **99** ... uložení testovacího vzorku do registru R99

5 **STO** **15** ... uložení testovacího vzorku do registru R15

RST **R/S** ... start programu

RCL **99** [0] ... kontrola registru R99, obsahuje správně 0

RCL **15** [0] ... kontrola registru R15, obsahuje správně 0

RCL **00** [0] ... kontrola registru R00, obsahuje správně 0

Note: Do registru R00 není sice smyčkou zapisováno, ale obsahuje čítač, který má na konci smyčky hodnotu 0 a tím je zajištěno jeho vynulování.

9A Zlatý řez phi

Symbol **phi**

Call up with buttons **2nd** **2nd** **R/S**

Tlačítko **phi** slouží k vyvolání konstanty "Zlatý řez phi", která má hodnotu $\phi = (1 + \sqrt{5})/2 = 1.618033988749894848$.

9B Dekadický mód DEC

Symbol **DEC**

Call up with buttons **2nd** **2nd** **0**

Instrukce **DEC** přepne mód displeje do dekadického zobrazení (implicitní mód kalkulátoru). Mantisa čísla se zobrazí i zadává v DEC kódu, číslice 0 až 9, a to včetně desetinných míst mantisy. Exponent je vždy dekadické číslo.

Uvedením prefixu **INV** před instrukcí **DEC** se aktivuje pomocný ladicí mód, kdy se v prvním řádku displeje zobrazí mantisa zobrazeného čísla v interním formátu, jako 16 hexadecimálních číslic. Tímto způsobem lze zobrazit i skrytá desetinná místa mantisy. První číslice zleva je nejvyšší číslice. Nejvýznamnější bit mantisy s hodnotou '1' je skrytý a je nahrazen znaménkovým bitem. Ladicí mód se vypne volbou módu zobrazení bez prefixu **INV**.

V technickém módu Eng je exponent násobkem čísla 3.

9C Inkrementace a dekrementace registru Inc

Symbol **Inc**

Call up with buttons **2nd** **2nd** **.**

Instrukce **Inc** inkrementuje (zvýší o 1) obsah datového registru R00 až R99, jehož číslo je uvedeno jako parametr instrukce. Uvede-li se před

instrukcí **Inc** prefix **INV**, provede se opačná operace - dekrementace registru (snížení o 1).

Uvedením kódu **Ind** za kódem instrukce **Inc**, ale před zadáním parametru, se instrukce změní na nepřímé adresování **Inc Ind** (kód 6B, viz [6B Nepřímá inkrementace/dekrementace registru Inc Ind](#)).

Podobnou funkci mají operace **Op 20** až **Op 3F**, ale ty se uplatní pouze na registry R00 až R15.

*Note: Na rozdíl od původního kalkulátoru TI-58/59, který používal BCD interpretaci čísel, kalkulátor ET-58 používá binární formát čísel. Následkem toho nedochází k automatickému zaokrouhlování výsledků operací na dekadické číslice. To se může projevit tak, že po větším množství opakovaných operací (např. tisíce instrukcí **Inc**) se naakumuluje odchylka od celých čísel tak, že nebude správně detekována rovnost celého čísla. Z toho důvodu je doporučeno při větším množství operací s celými čísly mezivýsledky zaokrouhlovat na celá čísla instrukcí **round**.*

9D Bitová inverze NOT

Symbol **NOT**

Call up with buttons **2nd** **2nd** **+/-**

Instrukcí NOT se provede bitová inverze čísla na displeji (registr X). Při bitové inverzi se změní hodnoty bitů 0 na 1 a hodnoty bitů 1 na 0.

číselná soustava - numeral system

U bitové inverze nelze rozlišit, s jak velkým operandem se má operace provést. To je ošetřeno podle aktuálně nastavené číselné soustavy. Nejdříve se provede operace změna znaménka a dekrementace čísla (snížení o 1). Je-li nastavena dekadická číselná soustava DEC nebo je-li výsledek operace kladný, další operace se neprovádí. V ostatních případech (jiná číselná soustava než 10 a záporné číslo) se výsledek

operace zamaskuje tak, aby se číslice vešly na displej.

Example:

číselná soustava - numeral system

DEC ... dekadická číselná soustava

1 2 3 NOT [-124] ... inverzí čísla 123 je číslo -124

HEX ... hexadecimální číselná soustava

1 2 3 NOT [FFFFFFFFFFFFEDC] ... inverzí čísla 0x123 je 0xEDC

9E Procenta %

Symbol **%**

Call up with buttons **2nd 2nd =**

Instrukce **%** provede operaci s procenty mezi prvním a druhým operandem. Jedná-li se o nejnižší úroveň výrazu, může být výpočet opakován pro jiný první operand opakováním stiskem **=**.

Instrukce procent **%** má dva způsoby použití. Lze ho uvést mezi prvním a druhým operandem, nebo použít operátory + - * : a instrukci procent **%** uvést za operandy, namísto rovníčka **=**.

Módy operací s procenty:

1) Spolu s operátorem součtu **+** přičte k základu daný počet procent.

1 2 3 + 4 5 % [178.35] ... $123 + 45\% = 123 + 123 \cdot 45 / 100 = 178.35$

2) Spolu s operátorem rozdílu **-** odečte od základu daný počet procent.

$$1\ 2\ 3\ -\ 4\ 5\ \% \quad [67.65] \quad \dots \quad 123 - 45\% = 123 - 123 \cdot 45/100 = 67.65$$

3) Spolu s operátorem násobení \times vypočte počet procent ze základu.

$$1\ 2\ 3\ \times\ 4\ 5\ \% \quad [55.35] \quad \dots \quad 123 \cdot 45\% = 123 \cdot 45/100 = 55.35$$

4) Spolu s operátorem dělení \div vypočte, kolik procent je ze základu.

$$4\ 5\ \div\ 1\ 2\ 3\ \% \quad [36.585\dots] \quad \dots \quad 45/123\ \% = 45/123 \cdot 100 = 36.585\dots\%$$

5) Procento $\%$ na pozici operátoru vypočte procenta ze základu.

$$1\ 2\ 3\ \% \ 4\ 5 = [55.35] \quad \dots \quad 123 \cdot 45\% = 123 \cdot 45/100 = 55.35$$

15. Speciální operace Op

Parametr instrukce **Op** se zadává ve formě 2 hexadecimálních číslic.

Op 00 Vymazání tiskových registrů 1 až 4

Operace **Op 00** vynuluje obsahy tiskových registrů 1 až 4. Vynulování má stejný význam jako naplnění registrů mezerami.

Na rozdíl od původního TI-58/59, tiskové registry nejsou u ET-58 sdíleny se zásobníkem operací. Jedná se o samostatné nezávislé registry a nejsou modifikovány jinými operacemi než **Op 00** až **Op 04**.

Op 01..04 Nastavení tiskového registru 1..4

Operace **Op 01** až **Op 04** uloží obsah displeje (registr X) do tiskového registru 1 až 4 a používají se k přípravě textu k vytištění na displej. Každý tiskový registr může obsahovat až 8 znaků. Jeden znak se zadává jako 2 číslice dekadického čísla v rozsahu 00 až 99, přičemž 00 se zobrazí jako mezera.

Editor čísel umožňuje zadat až 16 číslic (což odpovídá maximu 8 znaků). Při pozdějším zobrazení zobrazí pouze max. 14 číslic, to ale není na závadu, protože vnitřně si i nadále uchovává všech 16 číslic. Není-li zadáno všech 16 číslic, doplní se text zleva mezerami, až do maxima 8 znaků.

Na rozdíl od původního TI-58/59, tiskové registry nejsou u ET-58 sdíleny se zásobníkem operací. Jedná se o samostatné nezávislé registry a nejsou modifikovány jinými operacemi než **Op 00** až **Op 04**.

Podrobněji k tabulce znaků viz kapitola [Tabulka znaků](#).

Example "Hello World!":

1 0 3 Op 53 ... Načte text "Hello" (= číslo 40 69 76 76 79)

Op 01 ... Uloží text do tiskového registru 1

1 0 4 Op 53 ... Načte text "World" (= číslo 55 79 82 76 68)

0 1 ... Na konec textu přidá znak '!

Op 02 ... Uloží text do tiskového registru 2

Op 1A ... Vytiskne registry 1 a 2 do textu 1. řádku při zastavení

Op 1F ... Nastaví textový mód displeje

... na 1. řádku se zobrazí text "Hello World!"

Op 09 Načtení knihovního programu

Operace **Op 09** přenese vybraný knihovní program (vybraný instrukcí **Pgm**) do hlavní paměti. Přenosem se přepíše uživatelský program. Je-li knihovní program delší než kapacita hlavní paměti 1000 kroků (ve standardní knihovně to není žádný z programů), dojde k oříznutí délky programu.

Op 0A Výstup registrů 1 a 2 na 1. řádek za běhu

Operace **Op 0A** vypíše text z tiskových registrů 1 a 2 (připravených operacemi **Op 01** a **Op 02**) do 1. řádku displeje.

Jedná se o text zobrazený pouze za běhu programu. Zastavením programu je obsah textu vymazán, což je výchozí obsah textu za běhu programu.

Op 0B Výstup registru 1 s X na 1. řádek za běhu

Operace **Op 0B** vypíše text z tiskového registru 1 (připraveného operací **Op 01**) do levé půlky 1. řádku displeje. Do pravé půlky (tj. 8 znaků) vypíše aktuální obsah registru X (obsah displeje).

Jedná se o text zobrazený pouze za běhu programu. Zastavením programu je obsah textu vymazán, což je výchozí obsah textu za běhu programu.

Note: Dodatečnou změnou obsahu registru X se obsah displeje nezmění.

Example, průběžné zobrazení stavu programu:

RST **LRN** ... aktivuje režim programování

CLR **STO** **01** ... vynuluje datový registr R01

1 **0** **0** **Op** **53** ... načte text "Running" (= číslo 50 85 78 78 73 78 71)

Op **01** ... uloží text do tiskového registru 1

Lbi **=** ... návěští '=', začátek smyčky

Inc **01** **RCL** **01** ... inkrementuje registr R01

Op **0B** ... vypíše registr 1 s X na 1. řádek za běhu

GTO **=** ... opakuje smyčku

LRN ... deaktivuje režim programování

RST **R/S** ... spustí program

... [Running 123]

... program inkrementuje X a průběžně vypisuje stav na 1. řádku

Op 0C Výstup půl-registru 1 s X na 1. řádek za běhu

Operace **Op** **0C** vypíše text z první poloviny (vyšší číslice) tiskového registru 1 (připraveného operací **Op** **01**) do první čtvrtiny 1. řádku displeje. Do zbytku řádku (tj. 12 znaků) vypíše aktuální obsah registru X (obsah displeje).

Jedná se o text zobrazený pouze za běhu programu. Zastavením programu je obsah textu vymazán, což je výchozí obsah textu za běhu programu.

Note: Dodatečnou změnou obsahu registru X se obsah displeje nezmění.

Op 0D Výstup registrů 3 a 4 na 2. řádek za běhu

Operace **Op** **0D** vypíše text z tiskových registrů 3 a 4 (připravených operacemi **Op** **03** a **Op** **04**) do 2. řádku displeje.

Jedná se o text zobrazený pouze za běhu programu. Zastavením programu je obsah textu nahrazen znakem 'C', což je výchozí obsah textu za běhu programu.

Op 0E Výstup registru 3 s X na 2. řádek za běhu

Operace **Op 0E** vypíše text z tiskového registru 3 (připraveného operací **Op 03**) do levé půlky 2. řádku displeje. Do pravé půlky (tj. 8 znaků) vypíše aktuální obsah registru X (obsah displeje).

Jedná se o text zobrazený pouze za běhu programu. Zastavením programu je obsah textu nahrazen znakem 'C', což je výchozí obsah textu za běhu programu.

Note: Dodatečnou změnou obsahu registru X se obsah displeje nezmění.

Op 0F Výstup půl-registru 3 s X na 2. řádek za běhu

Operace **Op 0F** vypíše text z první poloviny (vyšší číslice) tiskového registru 3 (připraveného operací **Op 03**) do první čtvrtiny 2. řádku displeje. Do zbytku řádku (tj. 12 znaků) vypíše aktuální obsah registru X (obsah displeje).

Jedná se o text zobrazený pouze za běhu programu. Zastavením programu je obsah textu nahrazen znakem 'C', což je výchozí obsah textu za běhu programu.

Note: Dodatečnou změnou obsahu registru X se obsah displeje nezmění.

Op 10 Funkce sign

Operace **Op 10** provede s obsahem registru X (obsah displeje) znaménkovou operaci sign. Je-li obsah registru menší než 0, bude výsledkem operace číslo -1. Je-li obsah registru větší než 0, bude výsledkem +1. Je-li obsah registru 0, zůstane 0 i nadále.

Namísto **Op 10** lze stejně tak použít posloupnost **INV |xl**.

Op 11 Variace

Operace **Op 11** vypočte variace (tj. rozptyl) statistických dat zadaných pomocí instrukce **Stat**. K výpočtu používá obsahy datových registrů R01 až R06, které byly připraveny instrukcí **Stat**. Do registru T (pomocný registr, obsah se zobrazí tlačítkem **x<>t**) uloží variace proměnné X, podle vzorce $\text{var}X = \text{sum}(x^2)/N - (\text{sum}(x)/N)^2$. Do registru X (obsah displeje) uloží variace proměnné Y, podle vzorce $\text{var}Y = \text{sum}(y^2)/N - (\text{sum}(y)/N)^2$.

Example:

INV CMs ... vymaže registry statistiky R01...R06 a registry X a T

2 3 Stat 4 5 Stat 6 7 Stat ... vloží data pro Y (X bude 0, 1, 2)

Op 11 [322.666...] ... variace Y je 322.666...

x<>t [.6666...] ... variace X je 0.6666...

Example 2:

... statistické registry jsou naplněny daty z příkladu k **Op 12**

Op 11 [242.4722...]

x<>t [4.1822...]

Op 12 Koeficienty lineární regrese

Operace **Op 12** vypočte metodou nejmenších čtverců koeficienty lineární regresní přímky, která vznikla aproximací dvojic hodnot (X, Y), zadaných pomocí statistické funkce **Stat**. Regresní přímka má tvar $y = m \cdot x + b$. **Op 12** uloží do registru T (pomocný registr, obsah se zobrazí tlačítkem **x<>t**) koeficient 'm', tedy strmost přímky. Do registru X (obsah displeje) uloží koeficient 'b', tedy posun přímky ve směru Y. Koeficient strmosti $m = (\text{sum}(x \cdot y) - \text{sum}(x) \cdot \text{sum}(y)/N) / (\text{sum}(x^2) - \text{sum}(x)^2/N)$. Koeficient posunu $b = (\text{sum}(y) - m \cdot \text{sum}(x))/N$.

Example:

INV CMs ... vymaže registry statistiky R01...R06 a registry X a T

1 0 1 . 3 x<>t 6 0 9 Stat ... bod 1 (101.3, 609)

1 0 3 . 7 x<>t 6 2 6 Stat ... bod 2 (103.7, 626)

9 8 . 6 x<>t 5 8 6 Stat ... bod 3 (98.6, 586)

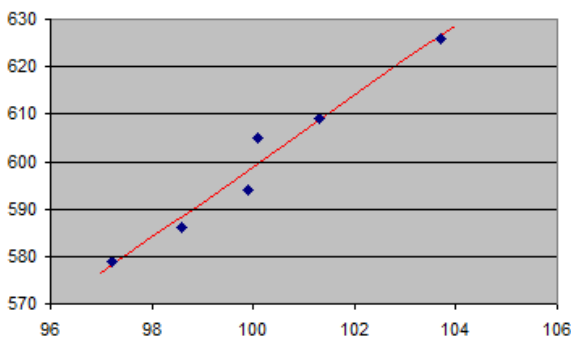
9 9 . 9 x<>t 5 9 4 Stat ... bod 4 (99.9, 594)

9 7 . 2 x<>t 5 7 9 Stat ... bod 5 (97.2, 579)

1 0 0 . 1 x<>t 6 0 5 Stat ... bod 6 (100.1, 605)

Op 12 [-148.506...] ... koeficient $b = -148.506...$

x<>t [7.4734...] ... koeficient $m = 7.4734...$



Op 13 Korelační koeficient

Korelační koeficient popisuje vzájemnou lineární závislost dvou veličin. Nabývá hodnot od -1 do +1. Hodnota -1 znamená, že Y je negativně lineárně závislé na X (zvýšením X se zmenší Y). Hodnota +1 představuje pozitivní lineární závislost. Hodnota 0 znamená, že veličiny nejsou na sobě závislé.

Operace **Op 13** vypočítá korelační koeficient z dat zadaných statistickou funkcí **Stat**. Korelační koeficient je počítán ze vztahu $R = m * \text{devx} / \text{devy}$. Koeficient strmosti regresní přímky $m = (\text{sum}(x*y) - \text{sum}(x)*\text{sum}(y)/N) / (\text{sum}(x^2) - \text{sum}(x)^2/N)$. Směrodatná odchylka pro X $\text{devx} =$

$\text{sqrt}((\text{sum}(x^2) - \text{sum}(x)^2/N)/(N-1))$. Směrodatná odchylka pro Y **devy** = $\text{sqrt}((\text{sum}(y^2) - \text{sum}(y)^2/N)/(N-1))$. Po dosazení a úpravě vztahů je výsledný vztah pro výpočet $R = (\text{sum}(x*y) - \text{sum}(x)*\text{sum}(y)/N) / \text{sqrt}((\text{sum}(x^2) - \text{sum}(x)^2/N)*(\text{sum}(y^2) - \text{sum}(y)^2/N))$.

Example:

... statistické registry jsou naplněny daty z příkladu k **Op 12**

Op 13 [0.0051370...]

Op 14 Lineární regrese Y z X

Operace **Op 14** vypočte hodnotu Y z hodnoty X pomocí lineární regresní přímky, podle vzorce $y = m*x + b$. Viz [Op 12 Koeficienty lineární regrese](#).

Example:

... statistické registry jsou naplněny daty z příkladu k **Op 12**

9 7 Op 14 [576.4166...] ... pro X=97 je Y=576.4166...

1 0 4 Op 14 [628.7306...] ... pro X=104 je Y=628.7306...

Op 15 Lineární regrese X z Y

Operace **Op 15** vypočte hodnotu X z hodnoty Y pomocí lineární regresní přímky, podle vzorce $x = (y - b)/m$. Viz [Op 12 Koeficienty lineární regrese](#).

Example:

... statistické registry jsou naplněny daty z příkladu k **Op 12**

5 8 0 Op 15 [97.4794...] ... pro Y=580 je X=97.4794...

6 3 0 Op 15 [104.170...] ... pro Y=630 je X=104.170...

Op 16, Op 17 Organizace paměti

Operace **Op 16** a **Op 17** slouží u původní TI-58/59 k nastavení předělu paměti RAM mezi pamětí programu a pamětí datových registrů. U ET-58 nelze předěl nastavovat a vždy odpovídá maximu 1000 programových kroků (v paměti EEPROM) a 100 či více datových registrů (v paměti RAM). Operace **Op 16** a **Op 17** pouze zobrazí číslo 999.99.

Op 18 Nastavení přepínače 7 bez chyby

Operace **Op 18** nastavení přepínač 7 v případě, že není indikována chyba E. V opačném případě zůstane stav přepínače 7 nezměněn. Stav přepínače lze testovat instrukcí **IfFlg**. Jinou možností testování stavu chyby je přepínač 0F, který je přímo spojen s indikací chyby E.

Op 19 Nastavení přepínače 7 při chybě

Operace **Op 19** nastavení přepínač 7 v případě, že je indikována chyba E. V opačném případě zůstane stav přepínače 7 nezměněn. Stav přepínače lze testovat instrukcí **IfFlg**. Jinou možností testování stavu chyby je přepínač 0F, který je přímo spojen s indikací chyby E.

Op 1A Výstup registrů 1 a 2 na 1. řádek při zastavení

Operace **Op 1A** vypíše text z tiskových registrů 1 a 2 (připravených operacemi **Op 01** a **Op 02**) do 1. řádku displeje.

Jedná se o text zobrazený, pokud program neběží a pokud je aktivní textový mód, zapnutý instrukcí **Op 1F**. Textový mód lze vypnout klávesou **CLR**.

Example - viz [Op 01...04 Nastavení tiskového registru 1..4](#)

Op 1B Výstup registru 1 s X na 1. řádek při zastavení

Operace **Op 1B** vypíše text z tiskového registru 1 (připraveného operací **Op 01**) do levé půlky 1. řádku displeje. Do pravé půlky (tj. 8 znaků) vypíše aktuální obsah registru X (obsah displeje).

Jedná se o text zobrazený, pokud program neběží a pokud je aktivní textový mód, zapnutý instrukcí **Op 1F**. Textový mód lze vypnout klávesou **CLR**.

Note: Dodatečnou změnou obsahu registru X se obsah displeje nezmění.

Op 1C Výstup půl-registru 1 s X na 1. řádek při zastavení

Operace **Op 1C** vypíše text z první poloviny (vyšší číslice) tiskového registru 1 (připraveného operací **Op 01**) do první čtvrtiny 1. řádku displeje. Do zbytku řádku (tj. 12 znaků) vypíše aktuální obsah registru X (obsah displeje).

Jedná se o text zobrazený, pokud program neběží a pokud je aktivní textový mód, zapnutý instrukcí **Op 1F**. Textový mód lze vypnout klávesou **CLR**.

Note: Dodatečnou změnou obsahu registru X se obsah displeje nezmění.

Op 1D Aktivace módu displeje 'Přepínače'

Operací **Op 1D** se aktivuje mód zobrazení 1. řádku displeje s přepínači. Jedná se o výchozí stav po zapnutí kalkulátoru. Je-li tento mód vybrán před aktivací módu 'Text' s **Op 1F**, je zpět obnoven stiskem tlačítka **CLR**.

Op 1E Aktivace módu displeje 'Registr T'

Operací **Op 1E** se aktivuje mód zobrazení 1. řádku displeje s registrem T. Na displeji se zobrazí 2 čísla (T a X) a to lze využít např. pro komplexní čísla. Je-li tento mód vybrán před aktivací módu 'Text' s **Op 1F**, je zpět obnoven stiskem tlačítka **CLR**.

Op 1F Nastavení módu displeje 'Text'

Operací **Op 1F** se aktivuje mód zobrazení 1. řádku displeje s textem. Text lze do řádku vytisknout pomocí operací **Op 1A** až **Op 1C**. Jedná se o text zobrazený pouze při zastavení chodu programu. Tlačítkem **CLR** lze textový mód displeje vypnout - navrátí se mód s přepínači **Op 1D** nebo s registrem T **Op 1E**, podle toho, který mód byl naposledy aktivní.

Vypnutím textového módu se obsah textového řádku nezmění. Obsah řádku lze předem připravit a zobrazit aktivací textového módu displeje až v případě potřeby.

Example - viz [Op 01...04 Nastavení tiskového registru 1..4](#)

Op 20 až Op 2F Inkrementace datového registru

Operace **Op 20** až **Op 2F** zvýší obsah datového registru R00 až R15 o 1 (inkrementace). Jedná se o operaci kompatibilní s původní TI-58/59. U ET-58 je vhodnější použít instrukci **Inc**, která obsluhuje všechny registry a umožňuje nepřímé adresování.

Op 30 až Op 3F Dekrementace datového registru

Operace **Op 30** až **Op 3F** sníží obsah datového registru R00 až R15 o 1 (dekrementace). Jedná se o operaci kompatibilní s původní TI-58/59. U ET-58 může být vhodnější použít instrukci **INV Inc**, která obsluhuje všechny registry a umožňuje nepřímé adresování.

Op 40 Vstup klávesy z klávesnice

Operace **Op 40** načte do registru X (číslo na displeji) kód stisknuté klávesy. Kód klávesy má hodnotu 0 až 254 a odpovídá kódu tlačítka v dekadické číselné soustavě. Není-li v bufferu klávesnice připraven žádný znak, je navraceno číslo 255. Kódy tlačítek zohledňují prefix **2nd**.

Example - výpis stisknutých kláves:

RST **LRN** ... aktivace programovacího módu

HEX ... přepnutí na HEX číselnou soustavu (pro snazší čitelnost kódů)

0F **0F** **x<>t** ... do registru T se připraví kód neplatného znaku 255=0xFF

Lbl **=** ... návěští začátku smyčky

Op **40** ... operace načtení klávesy z klávesnice

x=t **=** ... je-li navrácen kód 255, nic není stisknuto, návrat do smyčky

Pause ... probliknutí kódu stisknuté klávesy

GTO **=** ... pokračování ve smyčce

LRN ... ukončení programovacího módu

RST **R/S** ... start programu

... Po stisku tlačítka jeho HEX kód problikne na displeji (kromě tlačítek **2nd**, **R/S** a **GTO**). Konec programu s **R/S**.

Op 41 Test stisku tlačítka

Operace **Op** **41** umožňuje testovat, zda je stisknuto požadované tlačítko. Jako vstupní parametr operace se zadává nepřemapovaný kód tlačítka. V HEX kódu představuje první (vyšší) číslice řádek klávesnice 1 až 9, druhá (nižší) číslice je sloupec klávesnice 1 až 5. Kód tlačítka není přemapovaný prefixem **2nd**, tj. např. i číselné klávesy mají kód souřadnice (např. tlačítko **1** má HEX kód 82). Operace navrácí hodnotu 1, je-li testované tlačítko stisknuto. Není-li stisknuto, navrácí 0.

Example - test stisku tlačítka EE (zobrazí 1 je-li stisknuto):

RST **LRN** ... aktivace programovacího módu

HEX ... přepnutí na HEX číselnou soustavu

Lbl **=** ... návěští začátku smyčky

5 2 Op 41 ... test stisku **EE**, 5. řádek a 2. sloupec

Pause ... zobrazení stavu 1 (stisknuto) nebo 0 (nestisknuto)

GTO **=** ... pokračování ve smyčce

LRN ... ukončení programovacího módu

RST R/S ... start programu

... Zobrazuje 1 je-li **EE** stisknuto, jinak zobrazuje 0. Konec programu s **R/S**.

Op 42 Zobrazení jednoho znaku na displeji

Operace **Op 42** vypíše na displej 1 znak. Vypsání znaku probíhá do tiskových bufferů a zůstane tam dokud je platný příslušný tiskový buffer. Na vstupu operace je v registru X (obsah displeje) kód znaku k vypsání 00 až 99 (viz [Tabulka znaků](#)). V registru T je pozice na displeji 0 až 47.

Pozice 0 až 15 představuje 1. řádek displeje v případě, že program běží. Pozice 16 až 31 je 2. řádek displeje v případě, že program běží. Oba uvedené řádky se vynulují na implicitní hodnotu při zastavení programu.

Pozice 32 až 47 je 1. řádek v případě zastavení programu. Tento řádek je viditelný pouze v případě aktivace textového módu displeje s **Op 1F** (mód se vypne stiskem **CLR**).

Example:

RST LRN ... aktivace programovacího módu

Lbl A ... návěští podprogramu

1 5 x<>t ... do registru T se připraví pozice na konci 1. řádku když běží

1 0 Op 42 ... na konec řádku se vytiskne znak *

4 7 x<>t ... do registru T se připraví pozice konce 1. řádku když neběží

2 9 Op 42 ... na konec řádku se vytiskne znak =

Op 1F ... aktivuje se textový mód pro stav, když neběží

2 0 0 Op 44 ... prodleva 2 sekundy

RTN ... konec podprogramu (**INV SBR**)

LRN ... konec programovacího módu

A ... test programu. Po každém stisku běží asi 2 sekundy a po tu dobu zobrazuje na konci 1. řádku znak hvězdičky *. Když doběhne, zobrazí se na tom místě znak rovnítka =. Znak lze vymazat stiskem **CLR**.

Op 43 Načtení fontu

Kalkulátor umožňuje předefinovat 8 znaků LCD displeje, s kódem 92 až 99. Operace **Op 43** předefinuje znaky vybraným fontem podle čísla na displeji:

0 ... výchozí font

1 ... levý sloupec

2 ... pravý sloupec

3 ... linky a grafy

4 ... pixely

V případě fontu 1 až 4 je zpětné lomítko 60 \ nahrazeno svislou čarou |. Fonty lze zobrazit programem **A** knihovny ML-01.

Standardní výchozí font '0', znaky 00 až 99



Font pro levý sloupec '1', předdefinované znaky 92 až 99



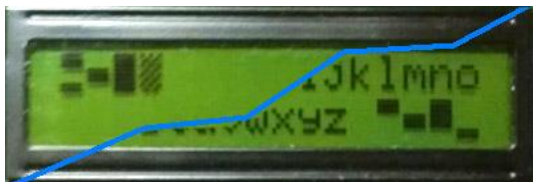
Font pro pravý sloupec '2', předdefinované znaky 92 až 99



Font pro linky a grafy '3', předdefinované znaky 92 až 99



Font pro pixely '4', předdefinované znaky 92 až 99



Example - zobrazení tabulek fontů:

Ovládání: **0** až **4** výběr fontu, **SST** a **BST** listování fontem, **R/S** konec.

RST **LRN** ... aktivace programovacího módu

CLR **STO** **01** ... příprava počátečního znaku do registru R01

Lbl **=** ... návěští začátku zobrazení jedné stránky znaků

3 2 **STO** **02** ... čítač zobrazených znaků do registru R02

CP ... příprava počáteční pozice znaku 0 do registru T

RCL **01** ... příprava prvního znaku k zobrazení

Lbl **x^2** ... návěští začátku smyčky zobrazení znaku

Op **42** ... zobrazení znaku X na pozici T

+ 1 = ... inkrementace znaku

x<>t + 1 = x<>t ... zvýšení pozice znaku

Dsz **2 x^2** ... smyčka zobrazení znaků jedné stránky

Lbl **Inx** ... začátek smyčky čekání na klávesu

Op 40 **x<>t** ... vstup znaku z klávesnice do registru T

6 5 INV x=t STO ... test klávesy SST

RCL 01 + 3 2 = ... zvýšení indexu stránky

AND 1 2 7 = STO 01 ... omezení indexu znaku

GTO = ... zobrazení nové stránky

Lbi STO ... návěští není-li SST

8 1 INV x=t RCL ... test klávesy BST

RCL 01 + 9 6 = ... snížení indexu stránky

AND 1 2 7 = STO 01 ... omezení indexu znaku

GTO = ... zobrazení nové stránky

Lbi RCL ... návěští není-li BST

4 INV x>=t Inx ... test klávesy 0..4

x<>t Op 43 ... načtení fontu 0..4

GTO = ... nové překreslení stránky

LRN ... ukončení programovacího módu

RST R/S ... start programu, ovládání **SST**, **BST**, **0...4**, konec **R/S**

Op 44 Prodleva

Operace **Op 44** čeká v programu po dobu 0 až 255 zadanou v registru X (číslo na displeji) jako násobek 10 ms (tj. max. 2.55 sekundy). Vlivem zpoždění při provádění programu může být výsledný čas o trochu delší než nastavený čas (o 10 až 20%).

Op 45 Zobrazení ukazatele zleva na 1. řádku za běhu

Operace **Op 45** načte do LCD displeje font číslo 1 (levý sloupec) a zobrazí na 1. řádku za běhu programu levý ukazatel (progress bar). Vstupem je hodnota X (obsah displeje) = 0 až 80.

Example - přibývající levý ukazatel na 1. řádku:

RST **LRN** ... aktivace režimu programování

CLR ... počáteční hodnota ukazatele = 0

Lbl **STO** ... návěští začátku cyklu

Op **45** ... zobrazení aktuálního ukazatele

+ **1** **=** ... zvýšení ukazatele

mod **8** **0** **=** ... přetečení ukazatele z 80 na 0

Op **80** ... krátká prodleva 10 ms

GTO **STO** ... opakování cyklu

LRN ... ukončení režimu programování

RST **R/S** ... start programu, ukazatel narůstá zleva doprava

Op 46 Zobrazení ukazatele zleva na 2. řádku za běhu

Operace **Op** **46** načte do LCD displeje font číslo 1 (levý sloupec) a zobrazí na 2. řádku za běhu programu levý ukazatel (progress bar). Vstupem je hodnota X (obsah displeje) = 0 až 80.

Example - přibývající levý ukazatel na 2. řádku:

RST **LRN** ... aktivace režimu programování

CLR ... počáteční hodnota ukazatele = 0

Lbl **STO** ... návěští začátku cyklu

Op **46** ... zobrazení aktuálního ukazatele

+ **1** **=** ... zvýšení ukazatele

mod **8** **0** **=** ... přetečení ukazatele z 80 na 0

Op **80** ... krátká prodleva 10 ms

GTO **STO** ... opakování cyklu

LRN ... ukončení režimu programování

RST R/S ... start programu, ukazatel narůstá zleva doprava

Op 47 Zobrazení textu s ukazatelem zleva při zastavení

Operace **Op 47** načte do LCD displeje font číslo 1 (levý sloupec), aktivuje textový mód displeje (**Op 1F**) a zobrazí na 1. řádce při zastavení programu levý ukazatel (progress bar). Vstupem je hodnota X (obsah displeje) = 0 až 40.

V následujícím příkladu se ukazatel zobrazuje za běhu programu instrukcí **Pause**, která zobrazuje obsah displeje stejně jako zastavený program.

Example - přibývající levý ukazatel s textem:

RST LRN ... aktivace režimu programování

1 0 6 Op 53 Op 01 ... načtení textu "Progress" do tiskového registru 1

CLR ... počáteční hodnota ukazatele = 0

Lbl STO ... návěští začátku cyklu

Op 47 ... zobrazení aktuálního ukazatele

+ 1 = ... zvýšení ukazatele

mod 4 0 = ... přetečení ukazatele ze 40 na 0

Pause ... krátká prodleva pro zobrazení 250 ms

GTO STO ... opakování cyklu

LRN ... ukončení režimu programování

RST R/S ... start programu, ukazatel narůstá zleva doprava

Op 48 Zobrazení ukazatele zprava na 1. řádce za běhu

Operace **Op 48** načte do LCD displeje font číslo 2 (pravý sloupec) a zobrazí na 1. řádce za běhu programu pravý ukazatel (progress bar).

Vstupem je hodnota X (obsah displeje) = 0 až 80.

Example - přibývající pravý ukazatel na 1. řádku:

RST **LRN** ... aktivace režimu programování

CLR ... počáteční hodnota ukazatele = 0

Lbl **STO** ... návěští začátku cyklu

Op **48** ... zobrazení aktuálního ukazatele

+ **1** **=** ... zvýšení ukazatele

mod **8** **0** **=** ... přetečení ukazatele z 80 na 0

Op **80** ... krátká prodleva 10 ms

GTO **STO** ... opakování cyklu

LRN ... ukončení režimu programování

RST **R/S** ... start programu, ukazatel narůstá zprava doleva

Op 49 Zobrazení ukazatele zprava na 2. řádku za běhu

Operace **Op** **49** načte do LCD displeje font číslo 2 (pravý sloupec) a zobrazí na 2. řádku za běhu programu pravý ukazatel (progress bar). Vstupem je hodnota X (obsah displeje) = 0 až 80.

Example - přibývající pravý ukazatel na 2. řádku:

RST **LRN** ... aktivace režimu programování

CLR ... počáteční hodnota ukazatele = 0

Lbl **STO** ... návěští začátku cyklu

Op **49** ... zobrazení aktuálního ukazatele

+ **1** **=** ... zvýšení ukazatele

mod **8** **0** **=** ... přetečení ukazatele z 80 na 0

Op 80 ... krátká prodleva 10 ms

GTO STO ... opakování cyklu

LRN ... ukončení režimu programování

RST R/S ... start programu, ukazatel narůstá zprava doleva

Op 4A Zobrazení textu s ukazatelem zprava při zastavení

Operace **Op 4A** načte do LCD displeje font číslo 2 (pravý sloupec), aktivuje textový mód displeje (**Op 1F**) a zobrazí na 1. řádku při zastavení programu pravý ukazatel (progress bar). Vstupem je hodnota X (obsah displeje) = 0 až 40.

V následujícím příkladu se ukazatel zobrazuje za běhu programu instrukcí **Pause**, která zobrazuje obsah displeje stejně jako zastavený program.

Example - přibývající pravý ukazatel s textem:

RST LRN ... aktivace režimu programování

1 0 6 Op 53 Op 01 ... načtení textu "Progress" do tiskového registru 1

CLR ... počáteční hodnota ukazatele = 0

Lbi STO ... návěští začátku cyklu

Op 4A ... zobrazení aktuálního ukazatele

+ 1 = ... zvýšení ukazatele

mod 4 0 = ... přetečení ukazatele ze 40 na 0

Pause ... krátká prodleva pro zobrazení 250 ms

GTO STO ... opakování cyklu

LRN ... ukončení režimu programování

RST R/S ... start programu, ukazatel narůstá zprava doleva



Op 4B Zobrazení sloupce grafu za běhu

Operace **Op 4B** načte do LCD displeje font číslo 3 (linky) a zobrazí za běhu programu sloupec grafu o hodnotě 0 až 16 podle registru X (na displeji) na pozici 0 až 15 podle registru T.

Example - animovaná pohybující se sinusovka

RST LRN ... aktivace režimu programování

CLR STO 01 ... počáteční fáze sinusovky 0 do registru 1

Deg ... úhel se bude počítat ve stupních

Lbl = ... návěští začátku zobrazení grafu

CP ... nulování pozice v registru T

Lbl + ... návěští začátku cyklu vykreslení 1 sloupceku

RCL 01 ... načtení fáze sinusovky

sin + 1 = * 8 = round ... hodnota sinusovky v rozsahu 0 až 16

Op 4B ... zobrazení jednoho sloupce grafu

RCL 01 + 2 2 | 5 = STO 01 ... zvýšení fáze sinusovky

X/T + 1 = X/T ... zvýšení pozice na displeji

1 6 INV x=t + ... test konce pozice, pokračování dalším sloupcem

RCL 01 + 2 2 | 5 = STO 01 ... zvýšení fáze sinusovky

GTO = ... animace sinusovky

LRN ... ukončení režimu programování

RST **R/S** ... start programu, sinusovka se pohybuje zprava doleva



Op 4C Zobrazení sloupce grafu s textem po zastavení

Operace **Op** **4C** načte do LCD displeje font číslo 3 (linky), aktivuje textový režim **Op** **1F** a zobrazí na 1. řádku po zastavení programu sloupec grafu o hodnotě 0 až 8 podle registru X (na displeji) na pozici 0 až 8 podle registru T, v pravé polovině 1. řádku, spolu s textem z tiskového registru 1 na levé polovině 1. řádku.

Example - sinusovka

RST **LRN** ... aktivace režimu programování

CLR **STO** **01** ... počáteční fáze sinusovky 0 do registru 1

1 **0** **7** **Op** **53** **0** **0** **Op** **01** ... načtení textu "Graph" do tiskového registru 1

Deg ... úhel se bude počítat ve stupních

CP ... nulování pozice v registru T

Lbl **+** ... návěští začátku cyklu vykreslení 1 sloupcečku

RCL **01** ... načtení fáze sinusovky

sin **+** **1** **=** ***** **4** **=** **round** ... hodnota sinusovky v rozsahu 0 až 8

Op **4C** ... zobrazení jednoho sloupce grafu spolu s textem

RCL **01** **+** **4** **5** **=** **STO** **01** ... zvýšení fáze sinusovky

X/T **+** **1** **=** **X/T** ... zvýšení pozice na displeji

8 **INV** **x=t** **+** ... test konce pozice, pokračování dalším sloupcem

R/S ... zastavení programu

LRN ... ukončení režimu programování

RST R/S ... start programu, zobrazí text "Graph" se sinusovkou



Op 4D Nastavení pixelu

Operace **Op 4D** načte do LCD displeje font číslo 4 (pixely) a nastaví za běhu na displeji pixel o souřadnici 0 až 15 horizontálně podle registru X (na displeji) a 0 až 5 vertikálně podle registru T.

Op 4E Vymazání pixelu

Operace **Op 4E** načte do LCD displeje font číslo 4 (pixely) a vymaže za běhu na displeji pixel o souřadnici 0 až 15 horizontálně podle registru X (na displeji) a 0 až 5 vertikálně podle registru T.

Op 4F Přepnutí pixelu

Operace **Op 4F** načte do LCD displeje font číslo 4 (pixely) a přepne (= zapne nebo vypne) za běhu na displeji pixel o souřadnici 0 až 15 horizontálně podle registru X (na displeji) a 0 až 5 vertikálně podle registru T.

Example - náhodné pixely:

RST LRN ... aktivace režimu programování

6 INV rand Int X/T ... náhodná vertikální souřadnice 0...5

1 6 INV rand Int ... náhodná horizontální souřadnice 0...15

Op 4F ... přepnutí pixelu

RST ... opakování

LRN ... ukončení režimu programování

RST **R/S** ... start programu, na displeji náhodně blikají pixely



Op 50 Vyhledání největšího společného dělitele

Operace **Op 50** vyhledá největší společný dělitel dvou celých nenulových čísel X (displej) a T, podle Euklidova algoritmu. Výsledek uloží do registru X (displej). Tato operace se používá ke krácení zlomků.

Example, krácení zlomku 260/340

2 6 0 X/T ... příprava prvního čísla do T registru

3 4 0 ... příprava druhého čísla do X registru

Op 50 [20] ... vyhledání největšího společného dělitele = 20

2 6 0 / 2 0 = [13] ... čitatel zlomku je 13

3 4 0 / 2 0 = [17] ... jmenovatel zlomku je 17

... zkrácený zlomek je 13/17.

Op 51 Načtení registru generátoru náhody

Operace **Op 51** navrátí v X (displej) hodnotu interního registru generátoru náhody (seed). Registr je celé číslo o velikosti DWORD a s rozsahem hodnot 0 až 4294967295.

Op 52 Nastavení registru generátoru náhody

Operace **Op 52** nastaví podle X (displej) hodnotu interního registru generátoru náhody (seed). Registr je celé číslo o velikosti DWORD a s rozsahem hodnot 0 až 4294967295. Nastavením registru lze dosáhnout reprodukovatelné náhodnosti.

Note: Registr generátoru náhody je při každém resetu kalkulátoru ukládán do EEPROM paměti a tím je zajištěna náhodnost generovaných hodnot téměř bez opakování i při opakovaných startech kalkulátoru. Nedoporučuje se registr generátoru náhody nastavovat, ztratila by se tím výhoda náhodnosti.

Op 53 Načtení předdefinovaného textu

Operace **Op 53** načte do registru X (displej) text předdefinovaný v interní tabulce procesoru. Délka textu je max. 8 znaků, tj. 16 číslic v interním formátu (viz [Tabulka znaků](#)). Před operací je potřeba do registru X (displej) zadat číslo textu podle následující tabulky. Za číslem textu je možné přidat desetinnou tečku a číslici desetin - v tom případě se načtený text posune doleva o počet pozic (mezer) daný číslicí desetin. Text je navrácen v editovatelném stavu - je možné k němu přidat další znaky. Example použití - viz [Op 01..04 Nastavení tiskového registru 1..4](#).

0 OK	30 Root	60 High	90 Calc
1 ERROR	31 Square	61 Too Low	91 Calcul
2 Diagnose	32 Key	62 Correct	92 Calculer
3 Result	33 Button	63 Too High	93 Won
4 Input	34 (1=YES)	64 Game	94 Wait
5 Enter	35 Vector	65 Time	95 Press
6 Index	36 Library	66 Working	96 a Key
7 Row	37 First	67 ...	97 Reaction
8 Column	38 Second	68 Win	98 Response
9 Continue	39 Third	69 Loss	99 Alien
10 STOP	40 Fourth	70 Bankroll	100 Running
11 Matrix	41 Param.	71 Success	101 Test
12 Complex	42 Entry	72 Crash	102 Help
13 Number	43 Output	73 Fail	103 Hello
14 Element	44 Rows	74 Speed	104 World
15 Item	45 Columns	75 Check	105 ET-58
16 Print	46 Size	76 Landing	106 Progress
17 from	47 Ready	77 Mission	107 Graph

18 to	48 Lambda	78 Complete	108 Black
19 Display	49 Polynom	79 Failure	109 White
20 Program	50 Angle	80 Smooth	110 Height
21 Load	51 Side	81 Turn	111 Width
22 Save	52 Triangle	82 Turns	112 CRC
23 YES	53 Area	83 Computer	
24 NO	54 Perimet	84 Your	
25 Add	55 Radius	85 You	
26 Subtract	56 ArcLen	86 My	
27 Multiply	57 Chord	87 Lost	
28 Divide	58 S-Area	88 Count	
29 Power	59 Low	89 Counter	

*Note: Operace **Op 53** ponechá číslo ve stavu zadávání číslic mantisy, kdy lze přidávat další znaky textu. Ovšem budete-li s číslem provádět další číselné operace (např. uložení do registru a zpětné načtení), může se číslo zobrazit v jiném tvaru, s desetinnými místy či s exponentem - v tom případě by nebylo možné pokračovat v přidávání znaků k mantise.*

Op 54 Přidání čísla k textu

Operací **Op 54** lze přidat k textu na displeji (v registru X) celé číslo se znaménkem z registru T. To slouží k přípravě textu označujícího např. číslo proměnné. Nelze takto dekodovat desetinné číslo. Editor čísla zůstane v editovatelném stavu, a tak lze přidávat další znaky.

*Note: Operace **Op 54** vypne mód exponentu EE i technický exponent Eng, protože exponent je s touto operací neslučitelný. Z textu mantisy odstraní případná desetinná místa vzniklá např. zobrazením se zaokrouhlením.*

Example výzvy "Enter a[i]:"

RST LRN ... aktivace módu programování

Lbl A ... návěští podprogramu pro zobrazení výzvy

X/T ... úschova indexu do registru T

5 Op 53 Op 01 ... načtení textu "Enter" do tiskového registru 1

6 5 5 9 ... text "a["

Op 54 ... přidání indexu z T

6 1 2 6 0 0 Op 02 ... přidání textu "]: " a uložení do tiskového registru 2

Op 1A ... výstup tiskových registrů do 1. řádku při zastavení

0 ... v editačním řádku se zobrazí 0

Op 1F ... zapnutí módu zobrazení textu

RTN ... konec podprogramu (**INV SBR**)

LRN ... ukončení programovacího módu

2 3 A ... test, na 1. řádku se zobrazí výzva "Enter a[23]."

Op 55 Inicializace zásobníku komplexních čísel a zlomků

Komplexní čísla a zlomky jsou dvojice čísel. Operandy se při výpočtu ukládají do zásobníku v datových registrech a zpracovávají se s využitím "Reverzní Polské Notace" RPN. To znamená, že se nejdříve do zásobníku čísel uloží operandy a potom se s nimi provede příslušná operace.

Operací **Op 55** se před prvním použitím komplexních čísel a zlomků nejdříve nadefinuje zásobník v datových registrech. V registru T je obsažen index prvního datového registru zásobníku, v registru X (na displeji) je počet čísel v zásobníku. Čísla se ukládají po párech, bude tedy potřeba dvojnásobek registrů, než je zadáný počet čísel. Implicitně (když se **Op 55** nepoužije) je zásobník nadefinován jako 10 čísel počínaje registrem R10, až po registr R29 (tedy jako když se zadá posloupnost **1 0 X/T 1 0 Op 55**).

Při výpočtech s komplexními čísly a se zlomky je doporučeno aktivovat kombinovaný mód displeje pomocí **Op 1E**. V tom případě se na 1. řádku displeje zobrazuje obsah registru T a na 2. řádku obsah registru X.

V případě výpočtů s komplexními čísly je v registru T obsažena reálná část čísla a v registru X (na displeji) imaginární část čísla. Po převodu na polární souřadnice obsahuje registr T modulus (absolutní hodnota čísla, poloměr) a registr X fázi (argument, úhel). Výpočty komplexních čísel se vždy provádí v kartézských souřadnicích. K převodu komplexního čísla mezi kartézskými a polárními souřadnicemi lze použít instrukci **P->R**.

V případě výpočtů se zlomky (a/b) obsahuje registr T čítelel 'a' a registr X jmenovatel 'b'. Ke krácení zlomků lze použít operaci **Op 50**, která vyhledá největší společný dělitel zlomku.

Op 56 Zjištění počtu čísel v zásobníku komplexních čísel

Operací **Op 56** je v X navracen počet čísel v zásobníku komplexních čísel a zlomků. Po inicializaci **Op 55** je navracena hodnota 0.

Op 57 Vložení čísla do zásobníku komplexních čísel

Operací **Op 57** je na vrchol zásobníku komplexních čísel a zlomků přidáno nové číslo. Na vstupu je v registru T reálná část komplexního čísla nebo čítelel zlomku, v registru X (displej) je imaginární část komplexního čísla nebo jmenovatel zlomku.

Op 58 Načtení čísla ze zásobníku komplexních čísel

Operací **Op 58** je z vrcholu zásobníku komplexních čísel a zlomků načteno číslo. Načtené číslo není ze zásobníku zrušeno, zásobník zůstane nezměněn. Na výstupu je v registru T navracena reálná část komplexního čísla nebo čítelel zlomku, v registru X (displej) je imaginární část komplexního čísla nebo jmenovatel zlomku.

Op 59 Zrušení čísla ze zásobníku komplexních čísel

Operací **Op 59** je z vrcholu zásobníku komplexních čísel a zlomků odstraněno poslední číslo. Počet čísel v zásobníku je operací snížen o 1.

Op 5A Záměna dvou čísel v zásobníku komplexních čísel

Operací **Op 5A** je v zásobníku komplexních čísel a zlomků zaměněno číslo na vrcholu zásobníku (poslední číslo) s předposledním číslem.

Op 5B Duplikace čísla v zásobníku komplexních čísel

Operací **Op 5B** je v zásobníku komplexních čísel a zlomků zduplikováno poslední číslo na vrcholu zásobníku. Počet čísel v zásobníku se operací zvýší o 1.

Op 5C Součet dvou komplexních čísel $X+Y$

Operace **Op 5C** přičte poslední komplexní číslo Y na vrcholu zásobníku k předposlednímu číslu X . Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek $X+Y$ a stane se novým posledním číslem na vrcholu zásobníku.

Op 5D Rozdíl dvou komplexních čísel $X-Y$

Operace **Op 5D** odečte poslední komplexní číslo Y na vrcholu zásobníku od předposledního čísla X . Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek $X-Y$ a stane se novým posledním číslem na vrcholu zásobníku.

Op 5E Násobek dvou komplexních čísel $X*Y$

Operace **Op 5E** vynásobí předposlední komplexní číslo X posledním číslem Y na vrcholu zásobníku. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek $X*Y$ a stane se novým posledním číslem na vrcholu zásobníku.

Op 5F Podíl dvou komplexních čísel X/Y

Operace **Op 5F** vydělí předposlední komplexní číslo X posledním číslem Y na vrcholu zásobníku. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek X/Y a stane se novým posledním číslem na vrcholu zásobníku.

Op 60 Umocnění dvou komplexních čísel X^Y

Operace **Op 60** umocní předposlední komplexní číslo X posledním číslem Y na vrcholu zásobníku. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek X^Y a stane se novým posledním číslem na vrcholu zásobníku.

Op 61 Odmocnění dvou komplexních čísel $X^{(1/Y)}$

Operace **Op 61** odmocní předposlední komplexní číslo X posledním číslem Y na vrcholu zásobníku. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek $X^{(1/Y)}$ a stane se novým posledním číslem na vrcholu zásobníku.

Op 62 Logaritmus dvou komplexních čísel $\log Y(X)$

Operace **Op 62** vypočte logaritmus předposledního komplexního čísla X se základem posledního čísla Y na vrcholu zásobníku. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek $\log Y(X)$ a stane se novým posledním číslem na vrcholu zásobníku.

Op 63 Druhá mocnina komplexního čísla X^2

Operace **Op 63** vypočte druhou mocninu komplexního čísla X na vrcholu zásobníku X^2 . Výsledek ponechá na pozici původního čísla X.

Op 64 Druhá odmocnina komplexního čísla \sqrt{X}

Operace **Op 64** vypočte druhou odmocninu komplexního čísla X na vrcholu zásobníku \sqrt{X} . Výsledek ponechá na pozici původního čísla X.

Op 65 Převrácená hodnota komplexního čísla $1/X$

Operace **Op 65** vypočte převrácenou hodnotu komplexního čísla X na vrcholu zásobníku $1/X$. Výsledek ponechá na pozici původního čísla X.

Op 66 Přirozený exponent komplexního čísla e^X

Operace **Op** **66** vypočte přirozený exponent komplexního čísla X na vrcholu zásobníku e^X . Výsledek ponechá na pozici původního čísla X .

Op 67 Přirozený logaritmus komplexního čísla $\ln(X)$

Operace **Op** **67** vypočte přirozený logaritmus komplexního čísla X na vrcholu zásobníku $\ln(X)$. Výsledek ponechá na pozici původního čísla X .

Op 68 Sinus komplexního čísla $\sin(X)$

Operace **Op** **68** vypočte sinus komplexního čísla X na vrcholu zásobníku $\sin(X)$. Výsledek ponechá na pozici původního čísla X .

Op 69 Kosinus komplexního čísla $\cos(X)$

Operace **Op** **69** vypočte kosinus komplexního čísla X na vrcholu zásobníku $\cos(X)$. Výsledek ponechá na pozici původního čísla X .

Op 6A Tangens komplexního čísla $\tan(X)$

Operace **Op** **6A** vypočte tangens komplexního čísla X na vrcholu zásobníku $\tan(X)$. Výsledek ponechá na pozici původního čísla X .

Op 6B Arkus sinus komplexního čísla $\operatorname{asin}(X)$

Operace **Op** **6B** vypočte arkus sinus komplexního čísla X na vrcholu zásobníku $\operatorname{asin}(X)$. Výsledek ponechá na pozici původního čísla X .

Op 6C Arkus kosinus komplexního čísla $\operatorname{acos}(X)$

Operace **Op** **6C** vypočte arkus kosinus komplexního čísla X na vrcholu

zásobníku $\text{acos}(X)$. Výsledek ponechá na pozici původního čísla X .

Op 6D Arkus tangens komplexního čísla $\text{atan}(X)$

Operace **Op 6D** vypočte arkus tangens komplexního čísla X na vrcholu zásobníku $\text{atan}(X)$. Výsledek ponechá na pozici původního čísla X .

Op 6E Konverze komplexního čísla na polární

Operace **Op 6E** zkonvertuje komplexní číslo X na vrcholu zásobníku z kartézských souřadnic na polární. Výsledek ponechá na pozici původního čísla X . S číslem v polárním tvaru nelze v zásobníku provádět aritmetické operace.

Op 6F Konverze polárního čísla na komplexní

Operace **Op 6F** zkonvertuje číslo X na vrcholu zásobníku z polárního tvaru na kartézské souřadnice. Výsledek ponechá na pozici původního čísla X . S číslem v polárním tvaru nelze v zásobníku provádět aritmetické operace.

Op 70 Vyhledání průchodu nulou funkce A'

Operace **Op 70** numericky hledá průchody nulou uživatelské funkce **A'**. Před použitím se v hlavním programu vytvoří funkce, označená návěštími **Lbl A'**, která vypočítá výstupní hodnotu y pro vstupní hodnotu x . Funkce **A'** nesmí používat rovnítko **=** ani mazat pomocí **CLR**.

Na vstupu operace **Op 70** je v registru T uložena koncová hodnota x , na které se vyhledávání nuly zastaví. V registru X bude výchozí vstupní hodnota. Funkce **Op 70** rozdělí interval mezi T a X na 100 úseků. V každém úseku testuje (opakovaným voláním uživatelské funkce **A'**), zda funkce prochází nulou, tj. zda se změní znaménko mezi začátkem a koncem úseku. Najde-li úsek s průchodem nulou, vyhledá přesné místo průchodu nulou, a to metodou půlení intervalů až do stavu, kdy je odchylka nepřesnosti mimo zobrazitelnou oblast.

Najde-li funkce průchod nulou, nalezenou hodnotu x navrátí v registru X. Nalezená hodnota x se může stát novou počáteční hodnotou pro nové hledání, protože funkce nehledá průchod nulou v počáteční hodnotě. Opětovným vyvoláním funkce **Op 70** se bude pokračovat v hledání dalšího průchodu nulou od poslední nalezené hodnoty x (která pro ten účel musí zůstat zachována v registru X).

Při hledání je potřeba dobře zvážit počátek a konec hledaného intervalu. Když je interval příliš velký, dělení na úseky může být příliš hrubé a průchod nulou se může minout. Naopak při příliš krátkém úseku může hledaný průchod nulou ležet mimo testovanou oblast.

Op 71 Simpsonův integrál funkce A'

Operace **Op 71** numericky vypočte integrál uživatelské funkce **A'**, Simpsonovou metodou. Před použitím se v hlavním programu vytvoří funkce, označená návěštím **Lbl A'**, která vypočítá výstupní hodnotu y pro vstupní hodnotu x. Funkce **A'** nesmí používat rovnítko **=** ani mazat pomocí **CLR**.

Před vyvoláním funkce se do HIR registru H1 uloží dolní hranice pro výpočet integrálu x0, do H2 se uloží horní hranice xn a do H3 se uloží počet kroků n (sudé číslo). Funkce navrátí hodnotu integrálu v registru X (na displeji).

Op 72 Konverze úhlu z aktuální úhlové jednotky na radiány

Operace **Op 72** zkonvertuje úhel z aktuální úhlové jednotky (vybrané uživatelem) na radiány. To umožňuje provádět goniometrické výpočty s úhly v radiánech, aniž je nutné měnit uživatelské nastavení úhlové jednotky.

Op 73 Konverze úhlu z radiánů na aktuální úhlovou jednotku

Operace **Op 73** zkonvertuje úhel z radiánů na aktuální úhlovou jednotku (vybranou uživatelem). To umožňuje provádět goniometrické výpočty s úhly v radiánech, aniž je nutné měnit uživatelské nastavení úhlové jednotky.

Op 74 Normální distribuce pravděpodobnosti Z(x)

Operace **Op 74** vypočítá normální distribuci pravděpodobnosti Z(x) pro hodnotu 'x' v rozsahu -150 až +150.

Op 75 Komplementární Gaussova distribuce Q(x) CGD

Operace **Op 75** vypočítá komplementární Gaussovu distribuci Q(x) (CGD, Q-funkce) pro hodnotu 'x' v rozsahu -8 až +8.

Op 76 Kumulativní normální distribuce P(x) CND

Operace **Op 76** vypočítá kumulativní normální distribuci P(x) (CND, CDF) pro hodnotu 'x' v rozsahu -8 až +8.

Op 77 Maximum

Operace **Op 77** porovná registry X (na displeji) a T, v registru X vrátí větší z hodnot.

Op 78 Minimum

Operace **Op 78** porovná registry X (na displeji) a T, v registru X vrátí menší z hodnot.

Op 79 Vynulování všech HIR registrů

Operace **Op 79** vynuluje obsahy všech HIR registrů H0 až H15.

Op 7A Konverze desetinného čísla na zlomek

Operace **Op 7A** zkonvertuje desetinné číslo 'x' na zlomek a/b, kdy v registru T navrátí číselník 'a' a v registru T navrátí jmenovatel 'b' zlomku. Funkce vynásobí desetinné číslo násobkem 518918400000, což je součin

prvočíslel: $2^{11} * 3^4 * 5^5 * 7 * 11 * 13$. Podaří-li se tímto způsobem dosáhnout celého čísla, vzniklý zlomek vykrátí největším společným dělitelem. Není-li desetinné číslo násobkem uvedených prvočísel, nemusí být nalezení zlomku úspěšné a může být navrácen čísel zlomku v desetinném tvaru.

Op 7B Převod zlomku na desetinné číslo

Operace **Op 7B** vydělí čísel zlomku 'a' jmenovatelem 'b' a výsledek navrátí v X jako desetinné číslo.

Op 7C Součet zlomků X+Y

Operace **Op 7C** přičte poslední zlomkové číslo Y na vrcholu zásobníku k předposlednímu číslu X. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek X+Y a stane se novým posledním číslem na vrcholu zásobníku.

Op 7D Rozdíl zlomků X-Y

Operace **Op 7D** odečte poslední zlomkové číslo Y na vrcholu zásobníku od předposledního čísla X. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek X-Y a stane se novým posledním číslem na vrcholu zásobníku.

Op 7E Násobek dvou zlomků X*Y

Operace **Op 7E** vynásobí předposlední zlomkové číslo X posledním číslem Y na vrcholu zásobníku. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek X*Y a stane se novým posledním číslem na vrcholu zásobníku.

Op 7F Podíl dvou zlomků X/Y

Operace **Op 7F** vydělí předposlední zlomkové číslo X posledním číslem Y na vrcholu zásobníku. Poslední číslo na vrcholu zásobníku zruší.

Předposlední číslo bude obsahovat výsledek X/Y a stane se novým posledním číslem na vrcholu zásobníku.

Op 80 Krátká prodleva 10 msec

Operace **Op 80** bude v programu čekat po dobu 10 msec (0.01 sekundy). Vlivem zpoždění při provádění programu může být výsledný čas o trochu delší než očekávaný čas (asi o 10 až 20%).

Op 81 Krátká prodleva 100 msec

Operace **Op 81** bude v programu čekat po dobu 100 msec (0.1 sekundy). Vlivem zpoždění při provádění programu může být výsledný čas o trochu delší než očekávaný čas (asi o 10 až 20%).

Op 82 Odstranění skrytých číslic

Operace **Op 82** odstraní skryté číslice čísla na displeji. Je to podobná funkce jako posloupnost **EE INV EE**, výhodou ovšem je, že není měněn mód exponentu. Tímto postupem lze snadno odstranit skryté číslice a zaokrouhlit číslo. Jinou alternativou je funkce **INV**, která ponechá číslo v editovatelném tvaru.

Op 83 Zahájení měření času

Kalkulátor používá vnitřní 16-bitový časovač s rozlišením 10 msec (0.01 sekundy) a délkou periody 10.9 minut. Operací **Op 83** se aktuální stav časovače uchová jako časová značka pro následná měření časového intervalu.

Op 84 Zjištění uplynulého času

Operace **Op 84** odečte od aktuální hodnoty čítače času časovou značku uchovanou operací **Op 83**. Rozdíl navrátí v registru X (displej) jako uběhlý čas v sekundách. Rozlišení získaného údaje je 10 msec (0.01 sekundy) a maximální měřitelná perioda je 10.9 minut. Po této periodě rozdíl času

přeteče a vynuluje se.

Metoda měření času pomocí interního čítače času je přesnější než funkce prodlevy (**Op 44**, **Op 80**, **Op 81**), protože nezávisí na zatížení procesoru při provádění programu.

Op 85 Zjištění počtu datových registrů

Operace **Op 85** navrátí počet datových registrů, které jsou k dispozici v kalkulátoru. To slouží k zajištění přenositelnosti programu mezi různými variantami kalkulátoru. Případně lze využít též k detekci varianty kalkulátoru.

Op 86 Zjištění stavu uživatelských přepínačů

Operace **Op 86** navrátí 16-bitové celé číslo (s hodnotou 0 až 65535), které představuje stav všech 16 uživatelských přepínačů. Funkce slouží k rychlému přístupu k přepínačům.

Op 87 Výpočet kontrolního součtu ROM paměti

Operace **Op 87** vypočte kontrolní součet ROM paměti kalkulátoru (metodou CRC-XModem) a navrátí ho v registru X jako celé 16-bitové číslo (rozsah 0 až 65535). Současně vrátí v registru T očekávanou hodnotu kontrolního součtu, uloženou v ROM. Pokud si údaje neodpovídají, kalkulátor je poškozen.

Op 88 Nastavení prodlevy pro vypnutí kalkulátoru

Operací **Op 88** lze nastavit dobu neaktivity, po které se kalkulátor sám vypne. Vstupem operace je doba v sekundách v registru X (displej), minimálně 5 sekund a maximálně 650 sekund (tj. téměř 11 minut). Zadáním hodnoty 0 se automatické vypínání kalkulátoru vyřadí z činnosti.

Op 89 Zjištění prodlevy pro vypnutí kalkulátoru

Operací **Op 89** lze zjistit dobu pro automatické vypnutí kalkulatoru při nečinnosti, v sekundách. 0 znamená vyžazené automatické vypínání.

Op 8A Zobrazení verze firmware kalkulatoru

Operace **Op 8A** zobrazí na displeji verzi firmware kalkulatoru stejně jako po resetu. Na displeji se na 1 sekundu zobrazí název varianty kalkulatoru spolu s 6-místným kódem, představujícím datum verze firmware kalkulatoru. Např. "ET-58 201005" znamená datum firmware (build) 5.10.2020.

Op 8B Reset kalkulatoru

Operace **Op 8B** resetuje kalkulator stejně, jako vyjmutí baterie.

16. Character Table

Tabulka znaků, použitá k tisku na displej, používá kódování znaků vycházející z ASCII kódu (sníženého o offset 32) a liší se od původní tabulky znaků TI-58/59. Znaky se zadávají jako 2 číslice dekadického čísla v rozsahu 00 až 99. Jeden datový registr může obsahovat až 8 znaků, to je 16 číslic. Pro ten účel umožňuje editor čísla zadat až 16 dekadických číslic, ovšem zobrazí pouze max. 14 číslic.

00 spc	16 0	32 @	48 P	64 '	80 p	96 odmocnina
01 !	17 1	33 A	49 Q	65 a	81 q	97 mikro
02 "	18 2	34 B	50 R	66 b	82 r	98 omega
03 #	19 3	35 C	51 S	67 c	83 s	99 obdélník
04 \$	20 4	36 D	52 T	68 d	84 t	
05 %	21 5	37 E	53 U	69 e	85 u	
06 &	22 6	38 F	54 V	70 f	86 v	
07 '	23 7	39 G	55 W	71 g	87 w	
08 (24 8	40 H	56 X	72 h	88 x	
09)	25 9	41 I	57 Y	73 i	89 y	
10 *	26 :	42 J	58 Z	74 j	90 z	
11 +	27 ;	43 K	59 [75 k	91 {	
12 ,	28 <	44 L	60 \	76 l	92	
13 -	29 =	45 M	61]	77 m	93 }	
14 .	30 >	46 N	62 ^	78 n	94 ~	
15 /	31 ?	47 O	63 _	79 o	95 pi	

Osm posledních znaků s kódem 92 až 99 lze přegenerovat načtením fontů pomocí operace **Op 43**. V případě přegenerování fontu je zpětné lomítko 60 \ nahrazeno svislou čarou |.

Example, text "Calculate filter"

C a l c u l a t e _ f i l t e r
 35 65 76 67 85 76 65 84 Op 01 69 00 70 73 76 84 69 82 Op 02

17. Knihovní programy

Kalkulátor ET-58 je vybaven knihovnou s bohatými knihovními programy. Programy knihovny se aktivují instrukcí **Pgm**, za kterou následuje číslo knihovního programu 1 až 50 (příp. u uživatelské knihovny více). Číslo 0 představuje hlavní uživatelský program. Knihovní programy lze prohlížet pomocí tlačítka **LRN** nebo přenést do hlavní paměti instrukcí **Op 09**.

Většina knihovních programů používá HIR registry (H0 až H15) jako pracovní registry. Pokud knihovna vyžaduje uživatelská data, přednostně používá datové registry od registru R10 výše, registry R00 až R09 rezervuje např. pro statistickou funkci **Stat**. Jen v některých případech využívá všechny datové registry.






U funkcí, které vypisují výzvu na 1. řádku displeje, lze text výzvy vymazat stiskem **CLR**. K resetování nastavení kalkulátoru do základního stavu lze použít funkci **Pgm 01 SBR CE**.

ML-01 Diagnostika

Build	Flags	Fill	Reset	
Font	Key	Stat	Clear	Diag

Knihovni program ML-01 slouží k diagnostickým a podpurným účelům.






Fonty



Zobrazení tabulky fontů. Na displeji se zobrazí stránka 32 znaků LCD displeje. Tlačítka  a  lze fontem listovat po stránkách. Font obsahuje 4 stránky se znaky 0 až 99 (poslední stránka je neúplná). Tlačítka  až  přepínají použitý font (viz [Op 43 Načtení fontu](#), spolu s ukázkami fontů): 0 implicitní, 1 levý sloupec, 2 pravý sloupec, 3 linky a grafy, 4 pixely. Funkce se ukončí tlačítkem .

Build


Zobrazení verze kalkulátoru (podobně jako po resetu). Na displeji kalkulátoru se na 2 sekundy zobrazí název varianty kalkulátoru spolu s 6-místným kódem, představujícím datum verze firmware kalkulátoru. Např. "ET-58 201005" znamená datum firmware (build) 5.10.2020. Na druhém řádku displeje se zobrazí kontrolní součet CRC paměti ROM.

Tlačítka HEX

Zobrazení kódu tlačítka. Funkce zobrazuje HEX kódy stisknutých tlačítek. Tlačítka jsou přemapovaná prefixem . Lze testovat všechna tlačítka kromě tlačítek ,  a . Funkce se ukončí tlačítkem .

Poznámka: Po přerušení funkce zůstane kalkulátor nastaven v HEX módu zobrazení. V tomto módu by další knihovní funkce nemusely pracovat správně, proto po ukončení funkce vraťte standardní dekadický mód instrukcí  nebo stiskem .

Tlačítka DEC

Zobrazení kódu tlačítka. Funkce zobrazuje DEC kódy stisknutých tlačítek. Tlačítka jsou přemapovaná prefixem .

kromě tlačítek **2nd**, **GTO** a **R/S**. Funkce se ukončí tlačítkem **R/S**.

Lbl C, Lbl CLR Nulování statistiky

Vynulování statistických registrů R01 až R06 (pro funkci **Stat**), vynulování registrů X a T.

Lbl C' Přepínače

Zobrazení stavu všech uživatelských přepínačů 0 až 15. Na 1. řádku displeje se zobrazí stav všech přepínačů jako skupina 16 číslic 0 a 1. Zobrazení lze ukončit stiskem **CLR**.

Lbl D, Lbl CE Nulování a inicializace

Funkce uvede kalkulátor do výchozího nastavení, vynuluje statistické registry R01 až R06, vynuluje registry X a T.

Lbl D' Naplnění registrů

Naplnění všech datových registrů číslem z displeje.

Lbl E, Lbl = Diagnostika

Funkce otestuje funkčnost kalkulátoru a též zkontroluje kontrolní součet CRC paměti ROM s firmware kalkulátoru. Podle výsledku testu zobrazí na displeji text buď "Diagnose OK" nebo "Diagnose ERROR".

Lbl E' Reset kalkulátoru

Funkce resetuje kalkulátor stejně, jako vyjmutí baterie.

ML-02 Determinant matice

i->x ->1/A j->1/a ->|A|.1/A
n j:a ->|A| i:b ->x

Program ML-02 slouží k výpočtu determinantu matice a její převrácené hodnoty. Podporuje čtvercové matice o rozměru 2x2 až 9x9.

Lb| A Rozměr matice

Zadání rozměru čtvercové matice 'n' (n = 1 ... 9).

Lb| B Zadání dat matice

Zadání indexu počátečního sloupce matice 'j' a zadávání dat matice. Data se zadávají od zadaného sloupce po řádcích shora dolů. Po zápisu každé položky se pokračuje stiskem **R/S**. V případě chyby lze znovu zadat číslo sloupce, stisknout **B** a pokračovat v zadávání od prvního řádku zadaného sloupce.

Lb| C Determinant

Výpočet determinantu matice.

Lb| D Zadání vektoru

Zadání indexu počáteční položky 'i' vektoru a zadávání dat vektoru. Po zápisu každé položky se pokračuje stiskem **R/S**. V případě chyby lze znovu zadat číslo počáteční položky, stisknout **D** a pokračovat v zadávání od zadané položky.

Lb| E Řešení rovnic

Řešení soustavy lineárních rovnic zadaných vektorem a maticí.

Lbl A' Čtení vektoru

Zadání indexu počáteční položky 'i' vektoru a čtení položek. Pokračování k další položce stiskem **R/S**. Čtení lze opakovat zadáním nového indexu položky a stiskem **A'**.

Lbl B' Inverze matice

Nejdříve musí být vypočítán determinant matice s **C**. Determinant nesmí být 0.

Lbl C' Čtení inverzní matice

Zadání indexu počátečního sloupce matice 'j' a čtení dat inverzní matice (po předešlém výpočtu s **B'**). Pokračování k další položce stiskem **R/S**. Data se čtou od zadaného sloupce shora dolů.

Lbl E' Determinant a inverze

Výpočet determinantu matice a inverze matice. Tato funkce v sobě zahrnuje volání funkcí **C** (determinant matice) a **B'** (inverze matice).

Example:

Mějme čtvercovou matici 3x3 s obsahem

$$A = \begin{pmatrix} 4 & 8 & 0 \\ 8 & 8 & 8 \\ 2 & 0 & 1 \end{pmatrix}$$

Pgm 02 ... aktivace knihovního programu ML-02

3 A ... zadání rozměru matice 3x3

1 B ... zahájení zadávání dat od 1. sloupce

4 R/S 8 R/S 2 R/S ... zadání dat 1. sloupce

8 R/S 8 R/S 0 R/S ... zadání dat 2. sloupce

0 **R/S** **8** **R/S** **1** **R/S** ... zadání dat 3. sloupce

C [96] ... výpočet determinantu, výsledek = 96

Hledáme takový vektor x , jehož násobkem $A \cdot x$ vznikne vektor b :

$$\mathbf{b} = \begin{pmatrix} 4 \\ 4 \\ 6 \end{pmatrix}$$

1 **D** ... budeme zadávat data výsledného vektoru od prvku 1

4 **R/S** **4** **R/S** **6** **R/S** ... zadání dat vektoru b

E ... řešení soustavy lineárních rovnic

1 **A'** ... budeme číst data vektoru x od prvku 1

R/S [4] **R/S** [-1.5] **R/S** [-2] ... řešení soustavy rovnic x

Řešením zadané soustavy rovnic je

$$\mathbf{x} = \begin{pmatrix} 4 \\ -1.5 \\ -2 \end{pmatrix}$$

Nyní chceme vypočítat inverzní matici A^{-1} . Determinant již máme vypočítaný z dříve.

B' ... výpočet inverzní matice

1 **C'** ... budeme číst data inverzní matice počínaje sloupcem 1

R/S [.0833...] **R/S** [.0833...] **R/S** [-.1666...] ... čtení sloupce 1

R/S [-.0833...] **R/S** [.0416...] **R/S** [.1666...] ... čtení sloupce 2

R/S [.6666...] **R/S** [-.3333...] **R/S** [-.3333...] ... čtení sloupce 2

Inverzní matice je (zaokrouhleno na 4 desetiny):

$$\mathbf{A}^{-1} = \begin{pmatrix} 0.0833 & -0.0833 & 0.6667 \\ 0.0833 & 0.04167 & -0.3333 \\ -0.1667 & 0.1667 & -0.3333 \end{pmatrix}$$

Případně můžeme ještě převést na zlomky pomocí ML-26:

$$\mathbf{A}^{-1} = \begin{pmatrix} 1/12 & -1/12 & 2/3 \\ 1/12 & 1/24 & -1/3 \\ -1/6 & 1/6 & -1/3 \end{pmatrix}$$

ML-03 Sčítání a násobení matic

j:c i:xi ->Ax i->y
m,n j:a j:b la1,la2 A+B

Program ML-03 umožňuje sčítání matic (s vynásobením skalární hodnotou) a násobení matic. Při sčítání se obě matice zadávají celé do paměti. Při násobení se zadává pouze sloupec druhé matice a také se čte pouze jeden sloupec, operace se opakuje postupně pro všechny sloupce.

Lbl A Rozměry matice

Zadání rozměrů matice: zadejte počet řádků 'm' a stiskněte **A**. Poté zadejte počet sloupců matice 'n' a stiskněte opět **A**.

Lbl B Zadání dat 1. matice

Zadejte index sloupce 1. matice, od kterého budete zadávat data ($j = 1 \dots n$) a stiskněte **B**. Poté zadávejte data první matice (A) po položkách sloupců v pořadí shora dolů. Pokračování stiskem **R/S**.

Lbl C Zadání matice pro sčítání

Zadejte index sloupce 2. matice pro sčítání, od kterého budete zadávat data ($j = 1 \dots n$) a stiskněte **C**. Poté zadávejte data druhé matice (B) po položkách sloupců v pořadí shora dolů. Pokračování stiskem **R/S**.

Lbl D Zadání skalárního násobku

Zadejte skalární násobek první matice lambda1 a stiskněte **D**. Poté zadejte skalární násobek druhé matice lambda2 a stiskněte opět **D**.

Lbl E Součet matic

Provedení součtu matic $C = \text{lambda1} * A + \text{lambda2} * B$

Lbl A' Čtení matice součtu

Zadejte index sloupce výsledné matice, od kterého budete číst data ($j = 1 \dots n$) a stiskněte **A'**. Čtete data po položkách sloupců v pořadí shora dolů. Pokračování stiskem **R/S**.

Lbl B' Zadání sloupce matice pro násobení

Zadejte číslo položky sloupce 2. matice pro násobení, od které budete zadávat data ($i = 1 \dots m$). Zadávejte data sloupce druhé matice, pokračujte s **R/S**. Násobení se provádí po sloupcích. Nejdříve zadáte sloupec 2. matice pro násobení, provedete násobení, přečtete výsledný sloupec a pak operaci opakuje postupně pro všechny sloupce druhé matice.

Lbl C' Násobení jednoho sloupce

Funkce **C'** provede vynásobení jednoho sloupce druhé matice s maticí A. Po přečtení výsledku lze pokračovat zadáním dalšího sloupce druhé matice.

Lbl D' Čtení sloupce výsledku násobení

Zadejte číslo položky sloupce výsledné matice, od které budete číst data ($i = 1 \dots n$). Čtete data a pokračujte s **R/S**. Po přečtení celého sloupce lze zadat další sloupec druhé matice a pokračovat násobením dalšího sloupce.

Example

Součet matic $C = A - 2*B$

A =	2	3	0
	1	0	5
B =	4	0	-1
	3	2	6

Pgm 03 ... aktivace knihovního programu ML-03

2 A 3 A ... zadání rozměru matice 2 řádky (m) a 3 sloupce (n)

1 **B** ... zadání dat první matice počínaje sloupcem 1

2 **R/S** **1** **R/S** ... zadání dat 1. sloupce

3 **R/S** **0** **R/S** ... zadání dat 2. sloupce

0 **R/S** **5** **R/S** ... zadání dat 3. sloupce

1 **C** ... zadání dat druhé matice počínaje sloupcem 1

4 **R/S** **3** **R/S** ... zadání dat 1. sloupce

0 **R/S** **2** **R/S** ... zadání dat 2. sloupce

1 **+/-** **R/S** **6** **R/S** ... zadání dat 3. sloupce

1 **D** **2** **+/-** **D** ... zadání lambda1 (1) a lambda2 (-2)

E ... výpočet součtu matic $C = A - 2 \cdot B$

1 **A'** ... zahájení čtení výsledku součtu počínaje sloupcem 1

[-6] **R/S** **[-5]** ... čtení dat 1. sloupce

R/S **[3]** **R/S** **[-4]** ... čtení dat 2. sloupce

R/S **[2]** **R/S** **[-7]** ... čtení dat 3. sloupce

Výsledná matice součtu

$$C = \begin{matrix} -6 & 3 & 2 \\ -5 & -4 & -7 \end{matrix}$$

Vynásobení výsledné matice C maticí D ($E = C \cdot D$)

$$D = \begin{matrix} 3 & 1 \\ 0 & 2 \\ 4 & 3 \end{matrix}$$

1 **B'** ... bude se zadávat sloupec 2. matice od 1. položky

3 **R/S** **0** **R/S** **4** **R/S** ... zadání dat 1. sloupce 2. matice

C' ... vynásobení 1. matice sloupcem 2. matice

1 **D'** ... bude se číst sloupec výsledku od 1. položky

[-10] **R/S** [-43] ... čtení 1. sloupce výsledku

1 **B'** ... bude se zadávat sloupec 2. matice od 1. položky

1 **R/S** **2** **R/S** **3** **R/S** ... zadání dat 2. sloupce 2. matice

C' ... vynásobení 1. matice sloupcem 2. matice

1 **D'** ... bude se číst sloupec výsledku od 1. položky

[6] **R/S** [-34] ... čtení 2. sloupce výsledku

Výsledná matice násobení

$$E = \begin{array}{cc} -10 & 6 \\ -43 & -34 \end{array}$$

ML-04 Komplexní aritmetika

INIT	X-Y	X:Y	logyX	X<>Y
ENTER	X+Y	XxY	X^Y	X^1/Y

Program ML-04 slouží k základním aritmetickým výpočtům s komplexními čísly. Komplexní čísla obsahují v registru T reálnou část čísla, v registru X (obsah displeje) imaginární část čísla. K operacím s komplexními čísly se používá zásobník 10 komplexních čísel, který je umístěn v datových registrech R10 až R29. Při operacích se nejdříve uloží do zásobníku potřebné operandy a poté se provede příslušná operace.

Lbl **A** Přidání čísla do zásobníku

Zadejte reálnou část čísla, stiskněte **X<>T**, zadejte imaginární část čísla a stiskněte **A**. Tím bylo zadané číslo přidáno na vrchol zásobníku.

Lbl **A'** Inicializace zásobníku

Instrukcí **A'** inicializujete zásobník čísel v registrech R10 až R29. Inicializaci můžete používat i opakovaně, budete-li chtít začít s novým výpočtem. Během inicializace se displej zapne do dvouřádkového módu. Na horním řádku se zobrazí reálná část komplexního čísla (registr T) a na dolním řádku imaginární část komplexního čísla (registr X). Dvouřádkový mód displeje lze ukončit operací **Op** **1D** nebo zavoláním podprogramu **Pgm** **01** **SBR** **CE**.

Lbl **B** Součet čísel X+Y

Instrukcí **B** se přičte číslo Y na vrcholu zásobníku k předposlednímu číslu X. Číslo Y z vrcholu zásobníku se zruší. Číslo X s výsledkem operace se stane novým číslem na vrcholu zásobníku. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl **B'** Rozdíl čísel X-Y

Instrukcí **B'** se odečte číslo Y na vrcholu zásobníku od předposledního čísla X. Číslo Y z vrcholu zásobníku se zruší. Číslo X s výsledkem operace

se stane novým číslem na vrcholu zásobníku. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl C Násobek čísel $X*Y$

Instrukcí **C** se číslem Y na vrcholu zásobníku vynásobí předposlední číslo X. Číslo Y z vrcholu zásobníku se zruší. Číslo X s výsledkem operace se stane novým číslem na vrcholu zásobníku. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl C' Podíl čísel $X:Y$

Instrukcí **C'** se číslem Y na vrcholu zásobníku vydělí předposlední číslo X. Číslo Y z vrcholu zásobníku se zruší. Číslo X s výsledkem operace se stane novým číslem na vrcholu zásobníku. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl D Mocnina čísla X^Y

Instrukcí **D** se umocní předposlední číslo X číslem Y na vrcholu zásobníku. Číslo Y z vrcholu zásobníku se zruší. Číslo X s výsledkem operace se stane novým číslem na vrcholu zásobníku. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl D' Logaritmus $\log Y(X)$

Instrukce **D'** vypočte logaritmus předposledního čísla X o základu čísla Y na vrcholu zásobníku. Číslo Y z vrcholu zásobníku se zruší. Číslo X s výsledkem operace se stane novým číslem na vrcholu zásobníku. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl E Odmocnina čísla $X^{(1/Y)}$

Instrukcí **E** se odmocní předposlední číslo X číslem Y na vrcholu zásobníku. Číslo Y z vrcholu zásobníku se zruší. Číslo X s výsledkem operace se stane novým číslem na vrcholu zásobníku. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl E' Záměna operandů

Instrukcí **E'** se zamění číslo Y na vrcholu zásobníku s předposledním číslem X. Současně se nové číslo z vrcholu zásobníku zobrazí na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Example, výpočet $((2 + 3i) * (1 - i))^{(1 + i)}$:

Pgm 04 ... aktivace knihovního programu ML-04

A' ... inicializace zásobníku komplexních čísel

2 **x<>t** **3** **A** ... uložení čísla $(2 + 3i)$ do zásobníku

1 **x<>t** **1** **+/-** **A** ... uložení čísla $(1 - i)$ do zásobníku

C ... vynásobení čísel $X * Y$

1 **x<>t** **1** **A** ... uložení čísla $(1 + i)$ do zásobníku

D ... mocnina $X ^ Y$

Výsledek výpočtu $(-1.0584... + 4.0495...i)$

ML-05 Komplexní funkce

INIT	e^X	x^2	P→R	DEL
ENTER	ln X	sqrt	R→P	1/X

Program ML-05 slouží k výpočtům funkcí s komplexními čísly. Komplexní čísla obsahují v registru T reálnou část čísla, v registru X (obsah displeje) imaginární část čísla. K operacím s komplexními čísly se používá zásobník 10 komplexních čísel, který je umístěn v datových registrech R10 až R29. Při operacích se nejdříve uloží do zásobníku potřebné operandy a poté se provede příslušná operace. Úhly jsou v radiánech.

Lbl **A** Přidání čísla do zásobníku

Zadejte reálnou část čísla, stiskněte **x<>t**, zadejte imaginární část čísla a stiskněte **A**. Tím bylo zadané číslo přidáno na vrchol zásobníku.

Lbl **A'** Inicializace zásobníku

Instrukcí **A'** inicializujete zásobník čísel v registrech R10 až R29. Inicializaci můžete používat i opakovaně, budete-li chtít začít s novým výpočtem. Během inicializace se displej zapne do dvouřádkového módu. Na horním řádku se zobrazí reálná část komplexního čísla (registr T) a na dolním řádku imaginární část komplexního čísla (registr X). Dvouřádkový mód displeje lze ukončit operací **Op** **1D** nebo zavoláním podprogramu **Pgm** **01** **SBR** **CE**.

Lbl **B** Přirozený logaritmus ln X

Instrukcí **B** se vypočte přirozený logaritmus čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl **B'** Přirozený exponent e^X

Instrukcí **B'** se vypočte přirozený exponent čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část

čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl C Odmocnina sqrt

Instrukcí **C** se vypočte druhá odmocnina čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl C' Druhá mocnina X^2

Instrukcí **C'** se vypočte druhá mocnina čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl D Převod na polární tvar R->P

Instrukcí **D** se číslo X na vrcholu zásobníku převede z kartézských souřadnic na polární tvar. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude modulus čísla (absolutní hodnota), na dolním řádku (registr X) bude argument čísla v radiánech (fáze, úhel). Další operace s číslem v polárních souřadnicích nejsou podporovány.

Lbl D' Převod z polárního tvaru P->R

Instrukcí **D'** se číslo X na vrcholu zásobníku převede z polárního tvaru (v radiánech) na kartézské souřadnice. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla. Operace s číslem v polárních souřadnicích nejsou podporovány.

Lbl E Převrácená hodnota $1/X$

Instrukcí **E** se vypočte převrácená hodnota čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část

čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl E' Zrušení čísla

Instrukcí **E'** se zruší číslo X na vrcholu zásobníku.

Example, výpočet odmocniny $\sqrt{2 + 3i}$:

Pgm 05 ... aktivace knihovního programu ML-05

A' ... inicializace zásobníku komplexních čísel

2 x<>t 3 A ... uložení čísla $(2 + 3i)$ do zásobníku

C ... výpočet odmocniny $\sqrt{}$

Výsledek výpočtu $(1.6714... + 0.89597...i)$

C' ... kontrolní výpočet x^2

Výsledek $(2 + 3i)$

ML-06 Komplexní trigonometrické funkce

INIT	asin X	acos X	atan X	ABS
ENTER	sin X	cos X	tan X	NEG

Program ML-06 slouží k výpočtům trigonometrických funkcí s komplexními čísly. Komplexní čísla obsahují v registru T reálnou část čísla, v registru X (obsah displeje) imaginární část čísla. K operacím s komplexními čísly se používá zásobník 10 komplexních čísel, který je umístěn v datových registrech R10 až R29. Při operacích se nejdříve uloží do zásobníku potřebné operandy a poté se provede příslušná operace. Úhly jsou v radiánech.

Lbl **A** Přidání čísla do zásobníku

Zadejte reálnou část čísla, stiskněte **x<>t**, zadejte imaginární část čísla a stiskněte **A**. Tím bylo zadané číslo přidáno na vrchol zásobníku.

Lbl **A'** Inicializace zásobníku

Instrukcí **A'** inicializujete zásobník čísel v registrech R10 až R29. Inicializaci můžete používat i opakovaně, budete-li chtít začít s novým výpočtem. Během inicializace se displej zapne do dvouřádkového módu. Na horním řádku se zobrazí reálná část komplexního čísla (registr T) a na dolním řádku imaginární část komplexního čísla (registr X). Dvouřádkový mód displeje lze ukončit operací **Op** **1D** nebo zavoláním podprogramu **Pgm** **01** **SBR** **CE**.

Lbl **B** Sinus X

Instrukcí **B** se vypočte sinus čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl **B'** Arkus sinus X

Instrukcí **B'** se vypočte arkus sinus čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace

zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl C Kosinus X

Instrukcí **C** se vypočte kosinus čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl C' Arkus kosinus X

Instrukcí **C'** se vypočte arkus kosinus čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl D Tangens X

Instrukcí **D** se vypočte tangens čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl D' Arkus tangens X

Instrukcí **D'** se vypočte arkus tangens čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl E Negace X

Instrukcí **E** se vypočte negace čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Lbl E' Absolutní hodnota X

Instrukcí **E'** se vypočte absolutní hodnota čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude reálná část čísla, na dolním řádku (registr X) bude imaginární část čísla.

Example, výpočet $\sinus(1 + 3i)$:

Pgm 06 ... aktivace knihovního programu ML-06

A' ... inicializace zásobníku komplexních čísel

1 **x<>t** **3** **A** ... uložení čísla $(1 + 3i)$ do zásobníku

B ... výpočet sinus

Výsledek výpočtu $(8.4716.. + 5.4126...i)$

B' ... kontrolní výpočet arkus sinus

Výsledek $(1 + 3i)$

ML-07 Vyčíslení polynomu

n i;ai x->P(x)

Program ML-07 počítá hodnotu polynomu $P(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n$, zadaného hodnotami koeficientů.

Lbl A Řád polynomu

Funkcí **A** se nastaví řád polynomu 'n'.

Lbl B Zadání koeficientů

Funkcí **B** se zadají hodnoty koeficientů polynomu. Vstupem funkce je index koeficientu $i=0..n$, od kterého budou hodnoty zadávány. Následuje zadání hodnot, pokračování klávesou **R/S**.

Lbl C Výpočet hodnoty

Funkcí **C** se vypočte hodnota polynomu pro zadanou hodnotu 'x'.

Example, polynom $P(x) = 2 - 3x + x^2$:

Pgm 07 ... aktivace knihovního programu ML-07

2 A ... nastavení řádu polynomu $n=2$

0 B ... zahájení zadávání koeficientů od indexu $i=0$

2 R/S 3 +/- R/S 1 R/S ... zadání koeficientů a_0 až a_2

2 C [0] ... výpočet hodnoty polynomu $P(2) = 0$

1 +/- C [6] ... výpočet hodnoty polynomu $P(-1) = 6$

ML-08 Průchod funkce nulou

ZERO

f(x) GRAPH

Program ML-08 numericky vyhledá průchody funkce nulou. To lze využít např. k vyhledání kořenů funkce, kterou je obtížné řešit obecně.

Jako první krok je potřeba v uživatelském hlavním programu vytvořit funkci označenou návěštím **[Lb]** **[A]**. Vstupem funkce je hodnota x , výstupem hodnota y . Funkce nesmí používat rovnítko **[=]** ani mazání s **[CLR]**.

[Lb] **[A]** Vyhledání nuly

Zadejte koncovou hodnotu ' x_2 ', do které maximálně má probíhat vyhledání nuly, a stiskněte **[x<>]**. Zadejte počáteční hodnotu ' x_1 ', od které bude probíhat vyhledávání, a stiskněte **[A]**. Po několika sekundách program zobrazí hodnotu ' x ' průchodu funkce nulou, nebo rozbliká 9.999+9999 jako indikaci že další průchod nenalezl. Stisknete-li znovu **[A]**, program vyhledá následující průchod nulou od zadaného místa, proto na displeji ponechte naposledy nalezenou hodnotu ' x '. V registru T stále zůstává zadaný horní limit intervalu.

Hledání nuly začíná od nepatrně vyšší hodnoty ' x ' než je zadaný počátek ' x_1 '. To proto, aby se opakovaně nenacházela naposledy nalezená hodnota x . Leží-li průchod nulou v počáteční hodnotě ' x_1 ', zvolte nižší výchozí ' x_1 '.

Program při hledání nuly nejdříve rozdělí zadaný interval x_1 až x_2 na 100 úseků. V každém úseku testuje, zda prochází nulou, tj. zda dojde ke změně znaménka hodnoty funkce. Nalezne-li úsek s průchodem nulou, vyhledá přesnou hodnotu místa ' x ' s průchodem nulou, metodou půlení intervalů. Vyhledání probíhá až do maximálního rozlišení, kdy se šířka testovaného úseku dostane mimo zobrazitelný údaj.

Funkce nenalezne průchod nulou v případě, kdy se křivka pouze dotýká osy ' x ' svým minimem či maximem, tj. nedojde ke změně znaménka v místě dotyku.

Začátek a konec testovaného intervalu je potřeba zvolit s rozvahou. Pokud se interval zvolí příliš velký, mohou některé průchody uniknout, protože v jednom 1/100 testovaném úseku může být více průchodů (úsek nevykáže změnu znaménka). Pokud se interval zvolí příliš malý, mohou naopak

některé průchody zůstat vně testované oblasti.

Lbl D Hodnota funkce

Funkce D vypočte hodnotu funkce A' v hlavním programu pro zadanou hodnotu x.

Lbl E Vykreslení grafu

Funkce **E** vykreslí na displeji graf funkce. Zadejte hodnotu 'x2' konce intervalu k vykreslení, stiskněte **x<>t**, zadejte hodnotu 'x1' začátku intervalu k vykreslení a stiskněte **E**. Zobrazení grafu lze ukončit stiskem kterékoliv klávesy (kromě **2nd**). Vykreslený graf je normalizován - je vyhledána maximální a minimální hodnota funkce f(x) a graf je roztažen na maximální rozsah hodnot.

Example, průchody nulou funkce $f(x) = 4*\sin(x) + 1 - x$:

RST LRN ... aktivace programovacího módu

Lbl A' ... označení návěští začátku funkce **A'**

(STO 10 ... úschova hodnoty 'x' do registru R10

sin x 4 ... $4*\sin(x)$

+ 1 - RCL 10) ... $+ 1 - x$

RTN ... konec podprogramu (**INV SBR**)

LRN ... ukončení programovacího módu

Pgm 08 ... aktivace knihovního programu ML-08

Rad ... osa 'x' bude představovat úhel zadaný v radiánech

3 x<>t 3 +/- ... bude se hledat v intervalu -3 až +3

A [-2.2100...] ... vyhledání prvního průchodu nulou = -2.2100...

A [-0.3421...] ... vyhledání druhého průchodu nulou = -0.3421...

A [2.7020...] ... vyhledání třetího průchodu nulou = 2.7020...

A [9.999+9999] ... další průchod nulou nenalezen

CLR ... zrušení indikace chyby

3 **x<>t** **3** **+/-** **E** ... vykreslení grafu v rozsahu -3 až +3



ML-09 Simpsonův integrál funkce

f(x)
x0 xn n ->I GRAPH

Program ML-09 počítá integrál spojitě funkce Simpsonovou aproximací.

Jako první krok je potřeba v uživatelském hlavním programu vytvořit funkci označenou návěštím **Lb| A'**. Vstupem funkce je hodnota x , výstupem hodnota y . Funkce nesmí používat rovnítko **=** ani mazání s **CLR**.

Lb| A Dolní limit x_0

Funkcí **A** se zadá dolní limit ' x_0 ' počítaného integrálu.

Lb| B Horní limit x_n

Funkcí **B** se zadá horní limit ' x_n ' počítaného integrálu.

Lb| C Počet kroků n

Funkcí **C** se zadá počet kroků ' n ', na které bude zadaný interval rozdělen. Počet kroků by mělo být sudé číslo. Není-li sudé číslo, program provede opravu na sudé číslo.

Lb| D Výpočet integrálu

Funkce **D** vypočte integrál uživatelské funkce **A'** v daném intervalu ' x_0 ' až ' x_n ' s daným počtem kroků ' n ', pomocí Simpsonovy aproximace.

Lb| E Graf

Funkce **E** vykreslí na displeji graf funkce v zadaném intervalu ' x_0 ' až ' x_n '. Zobrazení grafu lze ukončit stiskem kterékoliv klávesy (kromě **2nd**). Vykreslený graf je normalizován - je vyhledána maximální a minimální hodnota funkce $f(x)$ a graf je roztažen na maximální rozsah hodnot.

Lbl **A'** Hodnota funkce

Funkce **A'** vypočte hodnotu funkce **A'** v hlavním programu pro zadanou hodnotu x.

Example, integrál funkce $1/(\cos(x) + 2)$ v rozsahu 0 až $\pi/2$:

RST **LRN** ... aktivace programovacího módu

Lbl **A'** ... označení návěští začátku funkce **A'**

Rad ... osa 'x' bude představovat úhel zadaný v radiánech

(**cos** **+** **2** **)** **1/x** ... funkce $1/(\cos(x) + 2)$

RTN ... konec podprogramu (**INV** **SBR**)

LRN ... ukončení programovacího módu

Pgm **09** ... aktivace knihovního programu ML-09

0 **A** ... počátek intervalu 'x0'

pi **:** **2** **=** **B** ... konec intervalu $\pi/2$

2 **0** **C** ... počet kroků 'n'

D [0.6046...] ... výpočet integrálu

pi **B** **E** ... vykreslení grafu v intervalu 0 až π



2 ***** **pi** **=** **B** **+/-** **A** **E** ... vykreslení grafu v intervalu -2π až $+2\pi$



ML-10 Simpsonův diskrétní integrál

n dx i,fi ->I

Program ML-10 počítá numerický integrál z diskrétních hodnot Simpsonovou aproximací.

Lbl A Počet úseků 'n'

Funkcí **A** se zadá počet úseků 'n', na které je počítaný interval rozdělen. Musí se jednat o sudé číslo. Počet vstupních vzorků je 'n+1' (y0 až yn).

Lbl B Přírůstek dx

Funkcí **B** se zadá přírůstek souřadnice x, 'dx'.

Lbl C Zadání hodnot

Funkcí **C** se zadají hodnoty y, počínaje zadaným indexem i = 0...n. Pokračování klávesou **R/S**.

Lbl D Výpočet integrálu

Funkce **D** vypočítá hodnotu integrálu I pro zadané hodnoty y0...yn.

Example:

Pgm 10 ... aktivace knihovního programu ML-10

4 A ... počet úseků n = 4

1 B ... přírůstek dx = 1

0 C ... zahájení zadávání hodnot yi od i = 0

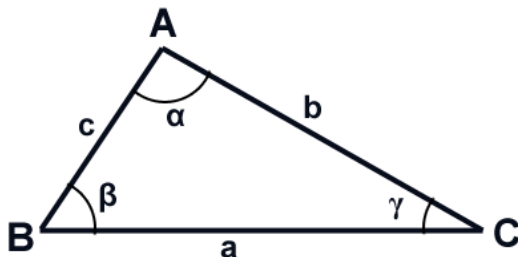
1 R/S 8 R/S 2 7 R/S 6 4 R/S 1 2 5 R/S ... zadání 5 hodnot

D [156] ... výpočet integrálu, I = 156

ML-11 Řešení trojúhelníků zadaných stranami

**SSS SSA ASS SAS PERIM
AREA**

Program ML-11 řeší trojúhelníky zadané převážně stranami (3 strany nebo 2 strany a 1 úhel).



Vrcholy trojúhelníku jsou označeny písmeny 'A', 'B', 'C'. Délky stran, nacházející se na protilehlé straně naproti vrcholu, jsou označeny 'a', 'b', 'c'. Úhly, svírané přilehlými stranami u vrcholu, jsou označeny 'α' (alfa), 'β' (beta) a 'γ' (gama). V programu jsou úhly označeny písmeny 'A', 'B' a 'C' (kalkulátor nemá k dispozici potřebná písmena řecké abecedy). S úhly se počítá v aktuálně zvolené úhlové míře.

U funkcí **A** až **D** se zadají známé vstupní parametry trojúhelníku. Po každé zadané hodnotě stisknete **R/S** pro pokračování. Kalkulátor vypočítá chybějící parametry a také obsah trojúhelníku a jeho obvod. Pokračování v zobrazení výsledků opět stiskem **R/S**. Ve funkci není nutné pokračovat pomocí **R/S**, lze kdykoliv zahájit jiný výpočet.

Lbl A Metoda SSS

Stisknete **A**, zadejte strany 'a', 'b' a 'c'. Program vypočítá úhly 'A', 'B', 'C', plochu trojúhelníku a jeho obvod. Pokračování klávesou **R/S**.

Lbl B Metoda SSA

Stisknete **B**, zadejte strany 'a', 'b' a úhel 'A'. Program vypočítá stranu 'c',

úhly 'B', 'C', plochu trojúhelníku a jeho obvod. Pokračování klávesou **R/S**.

Lbl **C** Metoda ASS

Stiskněte **C**, zadejte úhel 'B', strany 'a' a 'b'. Program vypočítá stranu 'c', úhly 'A', 'C', plochu trojúhelníku a jeho obvod. Pokračování klávesou **R/S**.

Lbl **D** Metoda SAS

Stiskněte **D**, zadejte stranu 'a', úhel 'C' a stranu 'b'. Program vypočítá stranu 'c', úhly 'A', 'B', plochu trojúhelníku a jeho obvod. Pokračování klávesou **R/S**.

Lbl **E** Plocha trojúhelníku

Funkce **E** vypočítá plochu trojúhelníku. Předtím musí být použita některá z funkcí **A** až **D**, čímž se vypočítají potřebné parametry trojúhelníku.

Lbl **E'** Obvod trojúhelníku

Funkce **E'** vypočítá obvod trojúhelníku. Předtím musí být použita některá z funkcí **A** až **D**, čímž se vypočítají potřebné parametry trojúhelníku.

Example:

Pgm **11** ... aktivace knihovního programu ML-11

Deg ... úhly jsou ve stupních

A ... výpočet metodou SSS

25 R/S 40 R/S 58 R/S ... zadání 3 stran 25, 40 a 58

[20.7509...] ... vypočtený úhel A = 20.7509...

R/S [34.5336...] ... vypočtený úhel B = 34.5336...

R/S [124.715...] ... vypočtený úhel C = 124.715...

R/S [410.99...] ... plocha 410.99...

R/S [123] ... obvod 123

B ... výpočet metodou SSA

3 **4** **6** **R/S** **5** **6** **R/S** **1** **5** **5** **R/S** ... zadání stran 3.46 a 5.6, úhel 15.5

[8.5159...] ... vypočtená strana c = 8.5159...

R/S [25.627...] ... vypočtený úhel B = 25.627...

R/S [138.87...] ... vypočtený úhel C = 138.87...

D ... výpočet metodou SAS

3 **8** **0** **R/S** **3** **8** **R/S** **3** **2** **0** **R/S** ... zadání strany 380, úhel 38 a strana 320

[234.85...] ... vypočtená strana c = 234.85...

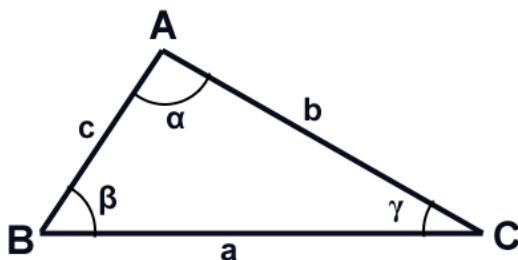
R/S [84.978...] ... vypočtený úhel A = 84.978...

R/S [57.021...] ... vypočtený úhel B = 57.021...

ML-12 Řešení trojúhelníků zadaných úhly

SAA AAS ASA AREA PERIM

Program ML-12 řeší trojúhelníky zadané převážně úhly (1 strana a 2 úhly).



Vrcholy trojúhelníku jsou označeny písmeny 'A', 'B', 'C'. Délky stran, nacházející se na protilehlé straně naproti vrcholu, jsou označeny 'a', 'b', 'c'. Úhly, svírané přilehlými stranami u vrcholu, jsou označeny 'α' (alfa), 'β' (beta) a 'γ' (gama). V programu jsou úhly označeny písmeny 'A', 'B' a 'C' (kalkulátor nemá k dispozici potřebná písmena řecké abecedy). S úhly se počítá v aktuálně zvolené úhlové míře.

U funkcí **A** až **C** se zadají známé vstupní parametry trojúhelníku. Po každé zadané hodnotě stisknete **R/S** pro pokračování. Kalkulátor vypočítá chybějící parametry a také obsah trojúhelníku a jeho obvod. Pokračování v zobrazení výsledků opět stiskem **R/S**. Ve funkci není nutné pokračovat pomocí **R/S**. Ize kdykoliv zahájit jiný výpočet.

Lb1 A Metoda SAA

Stisknete **A**, zadejte stranu 'a', úhly 'A' a 'B'. Program vypočítá strany 'b' a 'c', úhel 'C', plochu trojúhelníku a jeho obvod. Pokračování klávesou **R/S**.

Lb1 B Metoda AAS

Stisknete **B**, zadejte úhly 'A' a 'C', stranu 'a'. Program vypočítá strany 'b' a 'c', úhel 'B', plochu trojúhelníku a jeho obvod. Pokračování klávesou **R/S**.

Lbl **C** Metoda ASA

Stiskněte **C**, zadejte úhel 'B', stranu 'a', úhel 'C'. Program vypočítá strany 'b' a 'c', úhel 'B', plochu trojúhelníku a jeho obvod. Pokračování klávesou **R/S**.

Lbl **D** Plocha trojúhelníku

Funkce **D** vypočítá plochu trojúhelníku. Předtím musí být použita některá z funkcí **A** až **C**, čímž se vypočítají potřebné parametry trojúhelníku.

Lbl **E** Obvod trojúhelníku

Funkce **E** vypočítá obvod trojúhelníku. Předtím musí být použita některá z funkcí **A** až **C**, čímž se vypočítají potřebné parametry trojúhelníku.

Example:

Pgm **12** ... aktivace knihovního programu ML-12

Deg ... úhly jsou ve stupních

C ... výpočet metodou ASA

1 **1** **0** **R/S** **1** **8** **R/S** **5** **2** **.** **2** **R/S** ... zadání úhlu 110, strana 18, úhel 52.2

[55.331...] ... vypočtená strana b = 55.331...

R/S [46.526...] ... vypočtená strana c = 46.526...

R/S [17.8] ... vypočtený úhel A = 17.8

B ... výpočet metodou AAS

1 **7** **.** **8** **R/S** **8** **5** **.** **4** **R/S** **2** **.** **2** **5** **R/S** ... úhly 17.8 a 85.4, strana 2.25

[7.1658...] ... vypočtená strana b = 7.1658...

R/S [7.3365...] ... vypočtená strana c = 7.3365...

R/S [76.8] ... vypočtený úhel B = 76.8

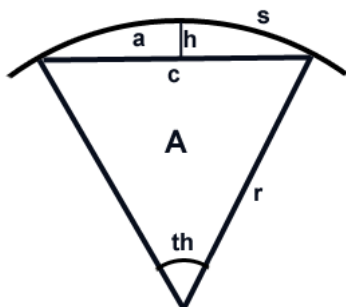
R/S [8.0355...] ... vypočtená plocha 8.0355...

R/S [16.7523...] ... vypočtený obvod 16.7523...

ML-13 Kruhový oblouk

th,r th,s th,c r,s r,c

Program ML-13 řeší kruhový oblouk, kruhovou výseč a kruhovou úseč.



th = středový úhel (théta θ)

r = poloměr kružnice

s = délka oblouku

c = délka tětiny (chord)

A = obsah kruhové výseče (celá plocha)

a = obsah kruhové úseče (část nad tětinou)

h = výška kruhové úseče

Lbl **A** Zadán úhel th a poloměr r

Po stisku **A** zadejte středový úhel 'th' a poloměr 'r'. Program vypočte délku oblouku 's', délku tětiny 'c', obsah výseče 'A', obsah úseče 'a' a výšku úseče 'h'. Pokračování stiskem **R/S**.

Lbl **B** Zadán úhel th a délka oblouku s

Po stisku **B** zadejte středový úhel 'th' a délku oblouku 's'. Program vypočte poloměr 'r', délku tětiny 'c', obsah výseče 'A', obsah úseče 'a' a výšku úseče 'h'.

'h'. Pokračování stiskem **R/S**.

Lbl **C** Zadán úhel θ a délka tětiny c

Po stisku **C** zadejte středový úhel ' θ ' a délku tětiny ' c '. Program vypočte poloměr ' r ', délku oblouku ' s ', obsah výseče ' A ', obsah úseče ' a ' a výšku úseče ' h '. Pokračování stiskem **R/S**.

Lbl **D** Zadán poloměr r a délka oblouku s

Po stisku **D** zadejte poloměr ' r ' a délku oblouku ' s '. Program vypočte středový úhel ' θ ', délku tětiny ' c ', obsah výseče ' A ', obsah úseče ' a ' a výšku úseče ' h '. Pokračování stiskem **R/S**.

Lbl **E** Zadán poloměr r a délka tětiny c

Po stisku **E** zadejte poloměr ' r ' a délku tětiny ' c '. Program vypočte středový úhel ' θ ', délku oblouku ' s ', obsah výseče ' A ', obsah úseče ' a ' a výšku úseče ' h '. Pokračování stiskem **R/S**.

Example:

Pgm **13** ... aktivace knihovního programu ML-13

Deg ... úhly jsou ve stupních

A **62** **R/S** **15** **R/S** ... zadán úhel $\theta = 62^\circ$ a poloměr $r = 15$

[16.2315...] ... délka oblouku $s = 16.2315...$

R/S [15.4511...] ... délka tětiny $c = 15.4511...$

R/S [121.736...] ... obsah výseče $A = 121.736...$

R/S [22.405...] ... obsah úseče $a = 22.405...$

R/S [2.1424...] ... výška úseče $h = 2.1424...$

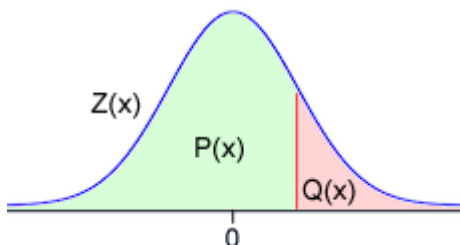
ML-14 Normální rozdělení

GRAPH
Z(x)

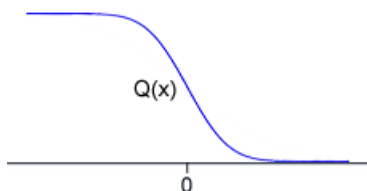
GRAPH
Q(x)

GRAPH
P(x)

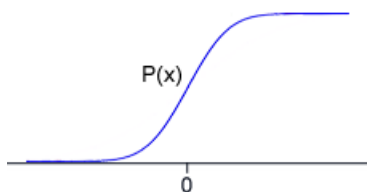
Program ML-14 počítá standardní normální (neboli Gaussovo) rozdělení pravděpodobnosti.



Křivka normální distribuce je popsána výrazem $Z(x) = 1/\sqrt{2\pi} \cdot \exp(-x^2/2)$.



Integrál (plocha) křivky od zadaného hraničního bodu $Q(x)$ se nazývá komplementární Gaussovo rozdělení, neboli též Q-funkce.



Integrál (plocha) křivky po zadaný hraniční bod $P(x)$ se nazývá kumulované normální rozdělení.

Lbl A Standardní rozdělení $Z(x)$

Funkce **A** počítá standardní normální rozdělení pravděpodobnosti $Z(x)$ pro 'x' v rozsahu -150 až +150. Na pozicích $x = -1$ a $+1$ se nacházejí inflekční body.

Lbl A' Graf standardního rozdělení

Funkce **A'** zobrazí graf standardního normálního rozdělení $Z(x)$. Ukončení zobrazení grafu stiskem kterékoliv klávesy (kromě **2nd**).

Lbl B Komplementární rozdělení $Q(x)$

Funkce **B** počítá komplementární rozdělení pravděpodobnosti $Q(x)$ pro 'x' v rozsahu -8 až +8.

Lbl B' Graf komplementárního rozdělení

Funkce **B'** zobrazí graf komplementárního rozdělení $Q(x)$. Ukončení zobrazení grafu stiskem kterékoliv klávesy (kromě **2nd**).

Lbl C Kumulované rozdělení $P(x)$

Funkce **C** počítá kumulované rozdělení pravděpodobnosti $P(x)$ pro 'x' v rozsahu -8 až +8.

Lbl C' Graf kumulovaného rozdělení

Funkce **C'** zobrazí graf kumulovaného rozdělení $P(x)$. Ukončení zobrazení grafu stiskem kterékoliv klávesy (kromě **2nd**).



ML-15 Generátor náhodných čísel

A,x B,s No(A,B) Int(A,B) No(x,s)

Program ML-15 používá ke generování náhodných čísel interní pseudonáhodný generátor kalkulátoru **rand**, založený na metodě LCG (Linear Congruential Generator, Lineární kongruentní generátor).

Lbl A Dolní hranice 'A' nebo střed 'x'

Funkce **A** slouží k zadání dolní hranice pro generování rovnoměrné náhodnosti 'A' nebo k zadání středu ke generování normální náhodnosti 'x'.

Lbl B Horní hranice 'B' nebo rozptyl 's'

Funkce **B** slouží k zadání horní hranice pro generování rovnoměrné náhodnosti 'B' nebo k zadání rozptylu ke generování normální náhodnosti 's'.

Lbl C Rovnoměrná náhodnost

Funkce **C** generuje náhodné desetinné číslo s rovnoměrným rozložením v intervalu od 'A' (včetně) do 'B' (vyjma).

Lbl D Rovnoměrná celočíselná náhodnost

Funkce **D** generuje náhodné celé číslo s rovnoměrným rozložením v intervalu od 'A' (včetně) do 'B' (vyjma).

Lbl E Normální náhodnost

Funkce **E** generuje náhodné desetinné číslo se standardním normálním (Gaussovým) rozložením se středem 'x' a rozptylem 's' podle vzorce $y = s * \sqrt{-2 * \ln(\text{rnd1})} * \cos(2 * \pi * \text{rnd2}) + x$.

ML-16 Variace, permutace, kombinace, faktoriál

big x!	log x!	sup x!	hyp x!	CLR
n!	ln n!	V rep	P rep	C rep
n	r	V	P	C

Program ML-16 slouží k výpočtům počtu variací, permutací, kombinací a faktoriálu. Variace znamená uspořádaný výběr 'r' prvků z 'n' množiny (na pořadí prvků záleží). Permutace je speciální případ variace, jsou-li vybrány všechny prvky ($r = n$). Kombinace znamená neuspořádaný výběr 'r' prvků z 'n' množiny (na pořadí prvků nezáleží). Výběry mohou být buď s možným opakováním prvků nebo bez opakování.

Lbl A Celkový počet prvků 'n'

Funkce **A** slouží k zadání celkového počtu prvků 'n'.

Lbl B Počet prvků výběru 'r'

Funkce **B** slouží k zadání počtu prvků výběru 'r'. Není zapotřebí při výpočtu permutace.

Lbl C Variace bez opakování

Funkce **C** vypočte počet variací (na pořadí záleží) výběru 'r' prvků z množiny 'n', bez opakování prvků.

Lbl D Permutace bez opakování

Funkce **D** vypočte počet permutací (na pořadí záleží) množiny 'n' prvků, bez opakování prvků.

Lbl E Kombinace bez opakování

Funkce **E** vypočte počet kombinací (na pořadí nezáleží) výběru 'r' prvků z množiny 'n', bez opakování prvků.

Lbl A' Faktoriál x!

Funkce **A'** vypočte faktoriál čísla na displeji 'x'. Faktoriál je násobek všech celých čísel od 1 po 'x'. Výpočet se provádí aproximací a číslem může být i desetinné číslo.

Lbl B' Přirozený logaritmus faktoriálu ln x!

Funkce **B'** vypočte přirozený logaritmus faktoriálu čísla na displeji 'x'. Faktoriál je násobek všech celých čísel od 1 po 'x'. Výpočet se provádí aproximací a číslem může být i desetinné číslo.

Lbl C' Variace s opakováním

Funkce **C'** vypočte počet variací (na pořadí záleží) výběru 'r' prvků z množiny 'n', s opakováním prvků.

Lbl D' Permutace s opakováním

Funkce **D'** vypočte počet permutací (na pořadí záleží) množiny 'n' prvků, s opakováním prvků.

Lbl E' Kombinace s opakováním

Funkce **E'** vypočte počet kombinací (na pořadí nezáleží) výběru 'r' prvků z množiny 'n', s opakováním prvků.

Lbl A'' Faktoriál velkých čísel x!

Funkce **A''** vypočte faktoriál velkých čísel 'x' na displeji, jejichž výsledek nelze zobrazit běžným způsobem (příliš velký exponent). Výsledek se zobrazí na 2 řádky displeje. Na horním řádku se nachází mantisa výsledku, na dolním řádku je dekadický exponent. Zobrazení lze přepnout na standardní tvar stiskem **E''** nebo **Op 1D**. Výpočet se provádí aproximací a číslem může být i desetinné číslo.

Lbl **B"** Dekadický logaritmus faktoriálu log x!

Funkce **B"** vypočte dekadický logaritmus faktoriálu čísla na displeji 'x'. Faktoriál je násobek všech celých čísel od 1 po 'x'. Výpočet se provádí aproximací a číslem může být i desetinné číslo.

Lbl **C"** Super-faktoriál sup x!

Funkce **C"** vypočte super-faktoriál čísla na displeji 'x', tedy násobek $n! = n! * (n-1)! * \dots$. Výsledek se zobrazí na 2 řádky displeje. Na horním řádku se nachází mantisa výsledku, na dolním řádku je dekadický exponent. Zobrazení lze přepnout na standardní tvar stiskem **E"** nebo **Op 1D**. Číslo musí být celé číslo (funkce číslo zaokrouhlí na celé číslo).

Lbl **D"** Hyper-faktoriál hyp x!

Funkce **D"** vypočte hyper-faktoriál čísla na displeji 'x', tedy násobek $H(n) = n^n * (n-1)^{(n-1)} * \dots$. Výsledek se zobrazí na 2 řádky displeje. Na horním řádku se nachází mantisa výsledku, na dolním řádku je dekadický exponent. Zobrazení lze přepnout na standardní tvar stiskem **E"** nebo **Op 1D**. Číslo musí být celé číslo (funkce číslo zaokrouhlí na celé číslo).

Lbl **E"** Reset zobrazení displeje

Funkce **E"** navrátí zobrazení displeje na standardní tvar s řádkem přepínačů, po funkcích **A"**, **C"** a **D"**. Lze též použít příkaz **Op 1D**.

*Note: Před následujícími příklady proved'te výběr knihovny **Pgm 16***

Poznámka: Při výpočtu faktoriálů velkých čísel se přechodně počítá logaritmus faktoriálu. Celá část představuje exponent výsledku, desetinná část obsahuje mantisu výsledku. Logaritmus faktoriálu je počítán s přesností 19 číslic. Po odstranění celočíselné části logaritmu zbude na desetinnou část méně číslic a tak se u velkých čísel může objevit chyba nepřesnosti až na pozici na displeji. Např. pro faktoriál čísla 123456789! má exponent 9 číslic. Na mantisu tak zbývá pouze 10 číslic a chyba se objeví na displeji, za desátou číslicí mantisy.

Example 1:

V policiče jsou 4 knihy. Kolik je možností uspořádání knih?

4 A ... zadání celkového počtu prvků $n = 4$

D [24] ... počet permutací bez opakování = 24

Example 2:

25 studentů se uchází o stipendium. Pouze 3 nejlepší mohou být vybráni a podle pořadí dostanou různou výšku stipendia. Kolik je možných výsledků?

2 5 A ... zadání celkového počtu prvků $n = 25$

3 B ... zadání počtu prvků výběru $r = 3$

C [13800] ... počet variací bez opakování = 13800

Example 3:

V balíčku je 52 karet. Hráč si vybere 4 karty. Kolik je možností výběru?

5 2 A ... zadání celkového počtu prvků $n = 52$

4 B ... zadání počtu prvků výběru $r = 4$

E [270725] ... počet kombinací bez opakování = 270725

Example 4:

V abecedě je 26 písmen. Kolik z nich můžeme sestavit slov o 3 písmenech?

2 6 A ... zadání celkového počtu prvků $n = 26$

3 B ... zadání počtu prvků výběru $r = 3$

C [17576] ... počet variací s opakováním = 17576

Example 5:

Kolika způsoby můžeme rozdělit 20 vstupenek mezi 10 studentů? Každý

student může dostat 0 až 20 vstupenek. Vybíráme tedy kombinace o 20 prvcích z opakující se množiny 10 prvků.

1 0 A ... zadání celkového počtu prvků $n = 10$

2 0 B ... zadání počtu prvků výběru $r = 20$

E [10015005] ... počet kombinací s opakováním = 10015005

Příklad 6:

Faktoriál velkého čísla 123456789!

1 2 3 4 5 6 7 8 9 A"

Výsledek: $123456789! = 2.853512521_{7299} \cdot 10^{945335859}$.

Poznámka: Jak lze zkontrolovat na stránce online kalkulátoru

<https://www.wolframalpha.com/input/?i=fakctorial+123456789> ,

mantisa výsledku je vypočtena s přesností jen 10 číslic. To je z toho důvodu, že funkce počítá nejdříve logaritmus faktoriálu s přesností 19 číslic. Celočíslná část logaritmu představuje 9 číslic exponentu. Desetinná část logaritmu obsahuje mantisu výsledku. Po odstranění 9 číslic exponentu zbude na mantisu 10 platných číslic (tedy poslední 4 číslice 7299 jsou již nesprávné).

Příklad 7:

Super-faktoriál čísla 100:

1 0 0 C"

Výsledek $100\$ = 2.7031768576518 \cdot 10^{6940}$.

Příklad 8:

Hyper-faktoriál čísla 25:

2 5 D"

Výsledek $H(25) = 4.9718576220366 \cdot 10^{386}$.

ML-17 Klouzavý průměr

n **m-AVG**

Program ML-17 počítá klouzavý průměr, tedy průměr z naposledy zadaných hodnot.

Lbl A Počet vzorků

Funkcí **A** se zadá počet prvků okna naposledy zadaných hodnot.

Lbl B Přidání vzorku

Funkcí **B** se přidá další hodnota a vypočte se aktuální klouzavý průměr.

Example:

Pgm 17 ... aktivace knihovního programu ML-17

3 A ... nastavení velikosti okna vzorků = 3

4 5 B [45] ... přidání vzorku 45, průměr = 45

5 0 B [47.5] ... přidání vzorku 50, průměr = 47.5

5 7 B [50.6666...] ... přidání vzorku 57, průměr = 50.6666...

6 5 B [57.3333...] ... přidání vzorku 65, průměr = 57.3333...

7 3 B [65] ... přidání vzorku 73, průměr = 65

8 1 B [73] ... přidání vzorku 81, průměr = 73

8 4 B [79.3333...] ... přidání vzorku 84, průměr = 79.3333...

8 4 B [83] ... přidání vzorku 84, průměr = 83

7 8 B [82] ... přidání vzorku 78, průměr = 82

6 8 B [76.6666...] ... přidání vzorku 68, průměr = 76.6666...

5 6 B [67.3333...] ... přidání vzorku 56, průměr = 67.3333...

ML-18 Složený úrok

S	(1+i)S	a	(1+i)a	INIT
N	%I	PV	FV	

Program ML-18 počítá složené úroky.

Lbl A Počet period N

Funkce **A** slouží k zadání počtu period N. Je-li zadána 0, počet period se vypočítá.

Lbl B Úroková sazba %I

Pomocí funkce **B** lze zadat úrokovou sazbu v % na periodu. Je-li zadána 0, úroková sazba se vypočítá.

Lbl C Současná hodnota PV

Pomocí funkce **C** se zadá současná hodnota PV (Present Value). Je-li zadána 0, současná hodnota se vypočítá.

Lbl D Budoucí hodnota FV

Pomocí funkce **D** se zadá budoucí hodnota FV (Future Value). Je-li zadána 0, budoucí hodnota se vypočítá.

Lbl A' Anuita pro amortizační fond Sni

Funkce **A'** řeší anuitu pro amortizační fond Sni.

Lbl B' Anuita pro budoucí rentu (1+i)Sni

Funkce **B'** řeší anuitu pro budoucí rentu/budoucí hodnotu FV (1+i)Sni.

Lbl C' Anuita pro současnou rentu ani

Funkce **C'** řeší anuitu pro současnou rentu/současnou hodnotu PV 'ani'.

Lbl D' Anuita pro budoucí rentu (1+i)ani

Funkce **D'** řeší anuitu pro budoucí rentu/současnou hodnotu PV (1+i)ani.

Lbl E' Inicializace

Funkce **E'** inicializuje program.

Example 1:

Pgm 18 ... aktivace knihovního programu ML-18

E' ... inicializace

2 4 A ... počet period = 24 měsíců

5 . 7 5 ... úroková sazba na 1 rok = 5.75%

: 1 2 = B [0.48] ... úroková sazba na 1 měsíc = 0.48%

5 0 0 C ... současná hodnota PV = 500

0 D [560.78] ... výpočet budoucí hodnoty FV za 12 měsíců = 560.78

Example 2:

Pgm 18 ... aktivace knihovního programu ML-18

E' ... inicializace

3 6 5 A ... počet period = 365 dnů

5 . 7 5 ... úroková sazba na 1 rok = 5.75%

: 3 6 5 = B [0.02] ... úroková sazba na 1 den = 0.02%

1 0 0 0 C ... současná hodnota PV = 1000

0 D [1059.18] ... výpočet budoucí hodnoty FV za 365 dnů = 1059.18

4 **A** ... nový počet period = 4 čtvrtletí

6 ... jiná úroková sazba na 1 rok = 6%

1 **4** **=** **B** [1.50] ... úroková sazba na 1 čtvrtletí = 1.50%

0 **D** [1061.36] ... jiná budoucí hodnota FV za 4 čtvrtletí = 1061.36

Example 3:

Pgm **18** ... aktivace knihovního programu ML-18

E ... inicializace

1 **2** **A** ... počet period = 12 měsíců

1 **C** ... současná hodnota PV = 1

1 **1** **0** **5** **7** **5** **D** ... budoucí hodnota FV = 1.0575 (navýšení o 5.75%)

0 **B** [0.47] ... výpočet potřebné úrokové sazby na 1 měsíc = 0.47%

2 **4** **A** ... počet period = 24 měsíců

5 **0** **0** **C** ... současná hodnota PV = 500

0 **D** [559.15] ... výpočet budoucí hodnoty FV za 24 měsíců = 559.15

Example 4:

Pgm **18** ... aktivace knihovního programu ML-18

E ... inicializace

1 **3** **A** ... počet period = 13 měsíců

1 **2** **3** **4** **C** ... současná hodnota PV = 1234

1 **3** **0** **0** **D** ... budoucí hodnota FV = 1300

0 **B** [0.40] ... výpočet potřebné úrokové sazby na 1 měsíc = 0.40%

1 **2** **A** ... počet period = 12 měsíců

1 **C** ... současná hodnota PV = 1

0 **D** [1.05] ... výpočet budoucí hodnoty FV = 1.05

1 **1** **=** **x** **1** **0** **0** **=** [4.93] ... navýšení je o 4.93%

ML-19 Splátky

sink	dueFV	ordPV	duePV	INIT
N	%I	PMT	PV/FV	B.PMT

Program ML-19 řeší splátky.

Lbl A Počet period N

Funkce **A** slouží k zadání počtu period N. Je-li zadána 0, počet period se vypočítá.

Lbl B Úroková sazba %I

Pomocí funkce **B** lze zadat úrokovou sazbu v % na periodu. Je-li zadána 0, úroková sazba se vypočítá.

Lbl C Platba na periodu PMT

Funkce **C** nastaví platbu na periodu PMT. Je-li zadána 0, platba na periodu se vypočítá.

Lbl D Současná nebo budoucí hodnota PV/FV

Funkcí **D** lze zadat současnou nebo budoucí hodnotu PV nebo FV. Zadáním 0 se položka vypočítá.

Lbl E Balonová (paušální) platba BAL

Funkcí **E** lze zadat balonovou platbu (větší paušální platba na konci období). Zadáním 0 se položka vypočítá.

Lbl A' Řešení potopného fondu 'sink'

Funkce **A'** řeší potopný fond (úspory k vyrovnaní dluhů, amortizační fond).

Lbl B' Řešení budoucí renty

Funkce **B'** řeší splátky budoucí renty FV.

Lbl C' Řešení důchodové renty

Funkce **C'** řeší běžnou anuitu důchodové renty.

Lbl D' Řešení spořicí renty

Funkce **D'** řeší anuitu spořicí renty.

Lbl E' Inicializace

Funkce **E'** inicializuje program.

Example 1, amortizační fond:

Pgm 19 ... aktivace knihovního programu ML-19

E' ... inicializace

A' ... bude se řešit amortizační fond

4 | 5 x ... počet let = 4.5

1 2 = A [54] ... počet měsíců celkem = 54

5 | 2 5 ... úroková sazba na 1 rok = 5.75%

: 1 2 = B [0.4375] ... úroková sazba na 1 měsíc = 0.4375%

2 5 C ... platba za periodu (měsíc) = 25

0 D [1519.08] ... výpočet budoucí hodnoty FV za 4.5 let = 1519.08

1 0 x 1 2 = A [120] ... nová doba 10 let = 120 měsíců

0 D [3934.42] ... výpočet budoucí hodnoty FV za 10 let = 3934.42

Example 2, budoucí renta:

Pgm **19** ... aktivace knihovního programu ML-19

E' ... inicializace

B' ... bude se řešit budoucí renta

10000D ... současná hodnota = 10000

50C ... platba (renta) za periodu (měsíc) = 50

10x12=A [120] ... doba 10 let = 120 měsíců

0B [0.7869] ... vypočtená renta = 78.69%

Example 3, důchodová renta

Pgm **19** ... aktivace knihovního programu ML-19

E' ... inicializace

C' ... bude se řešit důchodová renta

8.75 ... výše renty za rok = 8.75%

:12=B [0.7292] ... úroková sazba za 1 měsíc = 0.7292%

32000D ... budoucí hodnota FV = 32000

30x12=A [360] ... doba 30 let = 360 měsíců

0C [251.74] ... výpočet renty za 1 měsíc = 251.74

20x12=A [240] ... doba 20 let = 240 měsíců

0C [282.79] ... výpočet renty za 1 měsíc = 282.79

Example 4, spořicí renta

Pgm **19** ... aktivace knihovního programu ML-19

E' ... inicializace

D' ... bude se řešit spořicí renta

45000D ... budoucí hodnota FV = 45000

2000C ... splátka za 1 měsíc = 2000

1 0 0 0 0 E ... balonová platba BAL = 10000

2 x 1 2 = A [24] ... počet period = 24 měsíců

0 B [1.9638] ... výpočet úrokové sazby = 1.9638% na měsíc

x 1 2 = [23.5651] ... úroková sazba za 1 rok = 23.5651%

ML-20 Den v týdnu, dny mezi daty

(MMDD.YYYY)				AbsDay
Date1	Date2	NoDays	DayOfWeek	Julian

Program ML-20 počítá den v týdnu a počet dnů mezi dvěma daty.
Nejmenší podporované datum je 1.1.1583.

Lbl A Zadání data 1

Funkce **A** slouží k zadání prvního data, ve tvaru MMDD.RRRR (měsíc, den a rok).

Lb1 B Zadání data 2

Pomocí funkce `B` lze zadat druhé datum, ve tvaru MMDD.RRRR (měsíc, den a rok).

Lbl C Počet dnů mezi daty

Funkce C vypočte počet dnů mezi daty zadanými funkcemi A a B.

Lbl D Den v týdnu

Funkce `D` vypočte den v týdnu. Vstupem je datum ve tvaru MMDD.RRRR (měsíc, den a rok). Funkce vrací den v týdnu ve tvaru:

0 = sobota

1 = neděle

2 = pondělí

3 = úterý

4 = středa

5 = čtvrtek

6 = pátek

Lbl E Juliánský den

Funkce **E** zkonvertuje datum ve tvaru MMDD.RRRR (měsíc, den a rok) na Juliánský den, používaný v astronomii. Nejnižší podporované datum 1.1.1583 má Juliánský den 2299238. Juliánský den se mění v poledne, uváděný Juliánský den platí pro dopoledne (odpoledne je vyšší o 1).

Lbl E' Absolutní den

Funkce **E'** zkonvertuje datum ve tvaru MMDD.RRRR (měsíc, den a rok) na absolutní den, tj. počet dnů od počátku letopočtu. Nejnižší podporované datum 1.1.1583 má absolutní den 578179.

Example 1:

Pgm 20 ... aktivace knihovního programu ML-20

6 0 1 . 1 9 6 0 A ... zadání prvního data 1.6.1960

1 0 3 1 . 1 9 7 6 B ... zadání druhého data 31.10.1976

C [5996] ... počet dnů mezi daty = 5996

1 0 0 1 . 1 9 7 6 A ... nové první datum 1.10.1976

C [30] ... počet dnů mezi daty = 30

Example 2:

Pgm 20 ... aktivace knihovního programu ML-20

1 2 0 7 . 1 9 4 1 STO 01 ... datum 7.12.1941

D [1] ... dne 7.12.1941 byla neděle

RCL 01 E [2430335] ... Juliánské datum 2430335

ML-21 Hra HI-LO

**M INIT
START**

**M LO
GUESS**

**M HI
SCORE**

M CORR

Program ML-21 je hra HI-LO s hádáním čísel (menší-větší).

Lbl A Start nové hry

Funkce **A** spustí novou hru, kdy hráč hádá číslo.

Lbl B Hádání čísla

Zadejte hádané číslo 1 až 1023 a stiskněte **B**. Kalkulátor zobrazí -1 je-li váš odhad nízký, +1 je-li vysoko a rozblíká 0 při správné odpovědi.

Lbl C Skore

Funkce **C** zobrazí počet vašich pokusů.

Lbl A' Start hry počítače

Funkce **A'** spustí novou hru, kdy si hráč myslí číslo a kalkulátor hádá. Kalkulátor zobrazí hádané číslo - stiskněte **B'** je-li jeho odhad nízký, **C'** je-li jeho odhad vysoko a **D'** pokud se jeho odhad střelil.

Lbl B' Nízký odhad

Stiskněte **B'**, je-li odhad kalkulátoru nižší než vámi myšlené číslo.

Lbl C' Vysoký odhad

Stiskněte **C'**, je-li odhad kalkulátoru vyšší než vámi myšlené číslo.

Lbl D' Správný odhad

Stiskněte **D'**, je-li odhad kalkulátoru správný. Kalkulátor zobrazí počet pokusů.

ML-22 Běžný a spořicí účet

Checking Balance	Savings Deposit	I%/Yr Withdr	Periods/Yr No.Period	New Bal.
---------------------	--------------------	-----------------	-------------------------	----------

Program ML-22 slouží k vedení běžného a spořicího účtu. Typ účtu se volí funkcemi **A** a **B**. Po výběru typu účtu budou ostatní funkce nadále pracovat s vybraným účtem.

Lb **A** Balance

Funkce **A** navrátí aktuální bilanci účtu. Typ účtu se vybere klávesami **A** či **B**.

Lb **B** Vklad

Funkce **B** slouží k přidání vkladu na účet (zvýší bilanci účtu). Typ účtu se vybere klávesami **A** či **B**.

Lb **C** Výběr

Funkce **C** slouží k výběru z účtu (sníží bilanci účtu). Typ účtu se vybere klávesami **A** či **B**.

Lb **D** Počet period N

Funkce **D** slouží k zadání počtu uběhlých period N. Aktualizuje se balance spoření.

Lb **E** Nová balance

Funkce **E** nastaví novou bilanci účtu. Typ účtu se vybere klávesami **A** či **B**.

Lbl A' Výběr běžného účtu

Stiskem **A'** se bude nadále pracovat s běžným účtem.

Lbl B' Výběr spořicího účtu

Stiskem **B'** se bude nadále pracovat se spořicí účtem.

Lbl C' Úroková sazba l%

Funkcí **C'** lze zadat úrokovou sazbu l v % na rok.

Lbl D' Počet period na rok

Funkcí **D'** lze zadat počet sloučených period za rok.

Example:

Pgm 22 ... aktivace knihovního programu ML-22

A' ... výběr běžného účtu

231170E ... nastavení balance běžného účtu

231160B [463.30] ... vklad na běžný účet, nová balance = 463.30

50B [513.30] ... vklad na běžný účet, nová balance = 513.30

43110C [470.20] ... výběr z běžného účtu, nová balance = 470.20

18173C [451.47] ... výběr z běžného účtu, nová balance = 451.47

103179C [347.68] ... výběr z běžného účtu, nová balance = 347.68

10136C [337.32] ... výběr z běžného účtu, nová balance = 337.32

B' ... výběr spořicího účtu

1732184E ... nastavení balance spořicího účtu

5C' ... úroková sazba je 5% na rok

3 6 5 D' ... počet period na rok je 365 dnů

1 0 D [1735.22] ... posun o 10 dnů, nová balance = 1735.22

3 0 4 B [2039.22] ... vklad na spořicí účet, nová balance = 2039.22

4 D [2040.22] ... posun o 4 dny, nová balance = 2040.22

4 2 8 B [2468.33] ... vklad na spořicí účet, nová balance = 2468.22

6 D [2470.36] ... posun o 4 dny, nová balance = 2470.36

1 0 0 0 C [1470.36] ... výběr ze spořicího účtu, nová balance = 1470.36

1 0 D [1472.38] ... posun o 10 dnů, nová balance = 1472.38

ML-23 DMS operace

(dd.mmss)

n +- p * a / a

Program ML-23 slouží k výpočtům s časovými a úhlovými jednotkami. Údaje se zadávají ve tvaru dd.mmss, kdy před desetinnou tečkou je počet hodin nebo stupňů, 2 první číslice za desetinnou tečkou jsou minuty a další 2 číslice za desetinnou tečkou jsou vteřiny nebo sekundy. Obsahují-li vteřiny desetiny, jsou přidány jako další číslice za údajem vteřin.

Lb| A Zadání prvního čísla

Funkcí **A** lze zadat první číslo časového nebo úhlového údaje. Číslo se zadává ve formátu dd.mmss.

Lb| B Přičtení nebo odečtení druhého čísla

Funkcí **B** lze k prvnímu číslu přičíst nebo odečíst (podle znaménka) druhý časový údaj, také zadaný ve formátu dd.mmss. Má-li se výsledek operace použít k dalším operacím, je potřeba ho uložit opět jako první číslo klávesou **A**.

Lb| C Vynásobení čísla konstantou

Funkcí **C** lze první číslo vynásobit skalární konstantou. Má-li se výsledek operace použít k dalším operacím, je potřeba ho uložit opět jako první číslo klávesou **A**.

Lb| D Vydělení čísla konstantou

Funkcí **D** lze první číslo vydělit skalární konstantou. Má-li se výsledek operace použít k dalším operacím, je potřeba ho uložit opět jako první číslo klávesou **A**.

Example:

Pgm **23** ... aktivace knihovního programu ML-23

8 **A** ... první číslo = 8:00:00 (8 hodin)

3 **2** **B** [11.2000] ... přičtení času 3:20:00, výsledek 11:20:00

4 **7** **.** **0** **0** **3** **1** **A** ... první číslo = 47:00:31

2 **4** **.** **4** **3** **3** **5** **+/-** **B** [22.1656] ... odečtení 24:43:35, výsledek 22:16:56

2 **0** **.** **3** **0** **4** **5** **A** ... první číslo = 20:30:45

2 **C** [41.0130] ... vynásobení * 2, výsledek = 41.0130

1 **6** **0** **.** **8** **9** **7** **7** **A** ... první číslo = 160:89:77 (= 161:30:17)

2 **D** [80.4509] ... vydělení / 2, výsledek = 80:45:09

ML-24 Konverze jednotek 1

cm->in m->ft m->yd km->mi n mi->mi
in->cm ft->m yd->m mi->km mi->n mi

Program ML-24 slouží ke konverzi jednotek.

Lbl A Konverze palců na centimetry

Lbl A' Konverze centimetrů na palce

Lbl B Konverze stop na metry

Lbl B' Konverze metrů na stopy

Lbl C Konverze yardů na metry

Lbl C' Konverze metrů na yardy

Lbl D Konverze britské míle na kilometry

Lbl D' Konverze kilometrů na britskou míli

Lbl E Konverze britské míle na námořní míli

Lbl E' Konverze námořní míle na britskou míli

ML-25 Konverze jednotek 2

°C->°F lit->oz lit->gal grm->oz kg->lb
°F->°C oz->lit gal->lit oz->grm lb->kg

Program ML-25 slouží ke konverzi jednotek.

Lbl A Konverze °F na °C

Lbl A' Konverze °C na °F

Lbl B Konverze dutých uncí na litry

Lbl B' Konverze litrů na duté unce

Lbl C Konverze galonů na litry

Lbl C' Konverze litrů na galony

Lbl D Konverze uncí na gramy

Lbl D' Konverze gramů na unce

Lbl E Konverze liber na kilogramy

Lbl E' Konverze kilogramů na libry

ML-26 Aritmetika zlomků

INIT	a/b->d	X-Y	X:Y	DEL
ENTER	d->a/b	X+Y	XxY	X<>Y

Program ML-26 slouží k výpočtům se zlomky. Zlomky obsahují v registru T čítel, v registru X (obsah displeje) jmenovatel zlomku. K operacím se zlomky se používá zásobník 10 zlomkových čísel, který je umístěn v datových registrech R10 až R29. Při operacích se nejdříve uloží do zásobníku potřebné operandy a poté se provede příslušná operace.

Před výpočty se zlomky je potřeba nejdříve inicializovat zásobník čísel pomocí funkce **A'**. Funkce též přepne displej na dvouřádkový mód, kdy se v horním řádku displeje zobrazí čítel zlomku (registr T) a v dolním řádku jmenovatel zlomku (registr X). Ke zpětnému přepnutí displeje na standardní mód lze použít funkci **B'** nebo operaci **Op 1D** nebo program **Pgm 01 SBR CE**.

Lbl A' Přidání čísla do zásobníku

Zadejte čítel zlomku, stiskněte **x<>t**, zadejte jmenovatel zlomku a stiskněte **A'**. Tím bylo zadané číslo přidáno na vrchol zásobníku.

Lbl A' Inicializace zásobníku

Funkcí **A'** inicializujete zásobník čísel v registrech R10 až R29. Inicializaci můžete používat i opakovaně, budete-li chtít začít s novým výpočtem. Během inicializace se displej zapne do dvouřádkového módu. Na horním řádku se zobrazí čítel zlomku (registr T) a na dolním řádku jmenovatel zlomku (registr X). Dvouřádkový mód displeje lze ukončit funkcí **B'** nebo operací **Op 1D** nebo zavoláním podprogramu **Pgm 01 SBR CE**.

Lbl B Konverze desetín na zlomek

Funkcí **B** zkonvertuje desetinné číslo v registru X (displej) na zlomek a/b. Funkce vynásobí desetinné číslo násobkem 518918400000, což je součin prvočísel: $2^{11} * 3^4 * 5^5 * 7 * 11 * 13$. Podaří-li se tímto způsobem dosáhnout celého čísla, vzniklý zlomek vykrátí největším společným dělitelem. Není-li desetinné číslo násobkem uvedených prvočísel, nemusí

být nalezení zlomku úspěšné a může být navrácen čísel zlomku v desetinném tvaru. Funkce **B** současně přepne displej do dvouřádkového módu (je zobrazen současně registr T i X). Funkce pracuje s číslem na displeji, ne se zásobníkem.

Lb| B' Konverze zlomku na desetiny

Funkce **B'** vydělí čísel zlomku 'a' (v registru T) jmenovatelem 'b' (v registru X) a výsledek navrátí v X jako desetinné číslo. Funkce **B'** současně přepne displej do jednořádkového módu (je zobrazen registr X a příznaky). Funkce pracuje s číslem na displeji, ne se zásobníkem.

Lb| C Součet zlomků $X+Y$

Funkce **C** přičte poslední zlomkové číslo Y na vrcholu zásobníku k předposlednímu číslu X. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek $X+Y$ a stane se novým posledním číslem na vrcholu zásobníku. Výsledek operace normalizuje. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude čísel zlomku, na dolním řádku (registr X) bude jmenovatel zlomku.

Lb| C' Rozdíl zlomků $X-Y$

Funkce **C'** odečte poslední zlomkové číslo Y na vrcholu zásobníku od předposledního čísla X. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek $X-Y$ a stane se novým posledním číslem na vrcholu zásobníku. Výsledek operace normalizuje. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude čísel zlomku, na dolním řádku (registr X) bude jmenovatel zlomku.

Lb| D Násobek zlomků $X*Y$

Funkce **D** vynásobí předposlední zlomkové číslo X posledním číslem Y na vrcholu zásobníku. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek $X*Y$ a stane se novým posledním číslem na vrcholu zásobníku. Výsledek operace normalizuje. Současně je výsledek operace zobrazen na displeji - na horním řádku

(registr T) bude číselník zlomku, na dolním řádku (registr X) bude jmenovatel zlomku.

Lbl D' Podíl zlomků X:Y

Funkce **D'** vydělí předposlední zlomkové číslo X posledním číslem Y na vrcholu zásobníku. Poslední číslo na vrcholu zásobníku zruší. Předposlední číslo bude obsahovat výsledek X:Y a stane se novým posledním číslem na vrcholu zásobníku. Výsledek operace normalizuje. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude číselník zlomku, na dolním řádku (registr X) bude jmenovatel zlomku.

Lbl E Záměna operandů

Funkce **E** zamění číslo Y na vrcholu zásobníku s předposledním číslem X.

Lbl E' Zrušení čísla

Funkce **E'** zruší číslo X na vrcholu zásobníku.

Lbl +/- Negace X

Funkcí **SBR +/-** se vypočte negace čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude číselník zlomku, na dolním řádku (registr X) bude jmenovatel zlomku.

Lbl 1/x Převrácená hodnota 1/X

Funkcí **SBR 1/x** se vypočte převrácená hodnota čísla X na vrcholu zásobníku. Výsledek operace nahradí původní hodnotu čísla X. Současně je výsledek operace zobrazen na displeji - na horním řádku (registr T) bude číselník zlomku, na dolním řádku (registr X) bude jmenovatel zlomku.

Example, $23/6 + 1/3$:

Pgm **26** ... aktivace knihovního programu ML-26

A' ... inicializace zásobníku

2 **3** **x<>t** **6** **A** ... vložení prvního čísla, 23/6

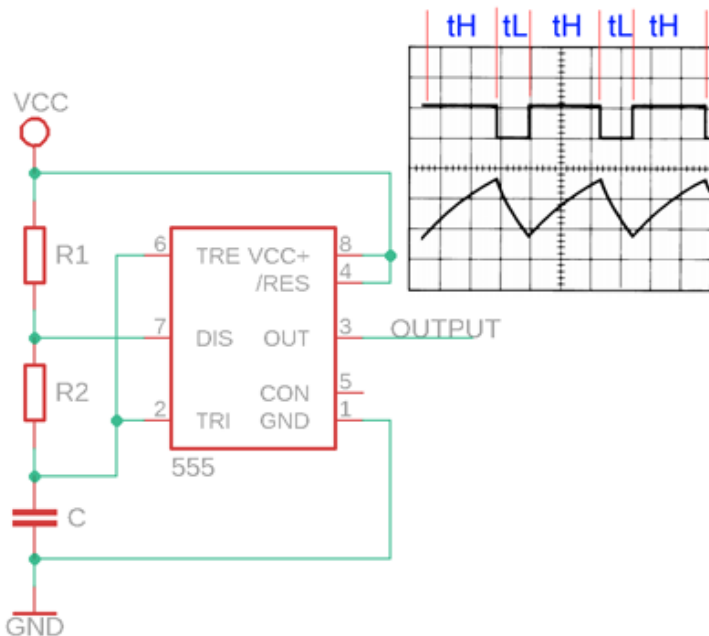
1 **x<>t** **3** **A** ... vložení druhého čísla, 1/3

C ... součet čísel, na displeji výsledek 25/6

ML-27 Astabilní generátor s obvodem 555

>C >R1 >R2 >f >tL
C uF R1 kΩ R2 kΩ f Hz >tH ms

Program ML-27 slouží k výpočtům obvodu 555 zapojeného jako astabilní generátor.



nabíjecí doba (výstup ve stavu HIGH) $t_H = \ln(2) * (R1 + R2) * C$

vybíjecí doba (výstup ve stavu LOW) $t_L = \ln(2) * R2 * C$

perioda $T = t_H + t_L = \ln(2) * (R1 + 2*R2) * C$

frekvence $f = 1/T$

střída signálu $D = t_H / T = (R1 + R2) / (R1 + 2*R2) = t_H * f$

Program počítá se 4 hlavními údaji: kondenzátor C (v uF), odpory R1 a R2 (v kohmech) a frekvence f (v Hz). Při výpočtech je potřeba znát 3 údaje, ze kterých program odvodí 4. údaj.

Lbl A Zadání kapacity C

Pomocí funkce **A** lze zadat kapacitu kondenzátoru C v uF.

Lbl A' Výpočet kapacity C

Funkce **A'** vypočítá kapacitu kondenzátoru C v uF z parametrů R1, R2 a f.

Lbl B Zadání odporu R1

Pomocí funkce **B** lze zadat hodnotu nabíjecího rezistoru R1 v kohmech.

Lbl B' Výpočet odporu R1

Funkce **B'** vypočítá odpor rezistoru R1 v kohmech z parametrů C, R2 a f.

Lbl C Zadání odporu R2

Pomocí funkce **C** lze zadat hodnotu vybíjecího rezistoru R2 v kohmech.

Lbl C' Výpočet odporu R2

Funkce **C'** vypočítá odpor rezistoru R2 v kohmech z parametrů C, R1 a f.

Lbl D Zadání frekvence f

Pomocí funkce **D** lze zadat frekvenci signálu f v Hz.

Lbl D' Výpočet frekvence f

Funkce **D'** vypočítá frekvenci signálu f v Hz.

Lbl E Výpočet doby tH

Funkce **E** vypočítá nabíjecí dobu tH (signál je ve stavu HIGH) v ms. Před

výpočtem musí být známy hodnoty C, R1 a R2.

LBL E' Výpočet doby tL

Funkce **E'** vypočítá vybíjecí dobu tL (signál je ve stavu LOW) v ms. Před výpočtem musí být známy hodnoty C a R2.

Example:

Pgm 27 ... aktivace knihovního programu ML-27

1 0 A ... známe kondenzátor C = 10 uF

1 B ... známe odpor R1 = 1 kohm

2 C ... známe odpor R2 = 2 kohm

D' [28.85...] ... výpočet frekvence f = 28.85... Hz

1/x [0.03465...] ... výpočet periody T = 34.65... ms

E [20.79...] ... výpočet doby tH = 20.79... ms

E' [13.86...] ... výpočet doby tL = 13.86... ms

E + E' = [34.65...] ... pro kontrolu, tH + tL = T

D' * E / 100 = [60] ... střída D = tH * f / 1000 * 100 = 60%

ML-28 (EE-07) Konverze poměrů

->P2/P1	->U2/U1	->Np	->dB
P2/P1	U2/U1	Np	dB

Program ML-28 konvertuje jednotky pro poměry, útlumy a zesílení. Zadáním údaje v jedné jednotce lze přechíst údaj zkonvertovaný na kteroukoliv jinou jednotku.

Lbl **A** Zadání poměru výkonů P2/P1

Lbl **A'** Výpočet poměru výkonů P2/P1

Lbl **B** Zadání poměru napětí U2/U1 nebo proudu I2/I1

Lbl **B'** Výpočet poměru napětí U2/U1 nebo proudu I2/I1

Lbl **C** Zadání poměru v Neperech Np

Lbl **C'** Výpočet poměru v Neperech Np

Lbl **D** Zadání poměru v decibelech dB

Lbl **D'** Výpočet poměru v decibelech dB

Example:

Pgm **28** ... aktivace knihovního programu ML-28

2 **0** **A** ... zadání poměru výkonů P2/P1 = 20

B' [4.472...] ... výpočet poměru napětí U2/U1 = 4.472...

C' [1.4978...] ... výpočet poměru v Neperech Np = 1.4978...

D' [13.010...] ... výpočet poměru v decibelech dB = 13.010...

ML-29 (EE-11) Reaktance LC

->f	->L	->C	XL->L	XC->C
f	L	C	->XL	->XC

Program ML-29 slouží k výpočtu reaktancí a rezonancí LC obvodů.

Lbl **A** Zadání frekvence f v Hz

Lbl **A'** Výpočet rezonanční frekvence f (L a C musí být známe)

Lbl **B** Zadání indukčnosti L v henry

Lbl **B'** Výpočet indukčnosti L v henry (f a C musí být známe)

Lbl **C** Zadání kapacity C ve faradech

Lbl **C'** Výpočet kapacity C ve faradech (f a L musí být známe)

Lbl **D** Výpočet induktivní reaktance X_L v ohmech (f a L musí být známe)

Lbl **D'** Přepočet induktivní reaktance X_L na indukčnost L (f musí být známe)

Lbl **E** Výpočet kapacitní reaktance X_C v ohmech (f a C musí být známe)

Lbl **E'** Přepočet kapacitní reaktance X_C na kapacitu C (f musí být známe)

Example:

Pgm **29** ... aktivace knihovního programu ML-29

Eng ... technický exponent

1 **5** **EE** **1** **2** **+/-** **C** ... zadání kapacity 15 pF

2 **7** **EE** **6** **A** ... zadání frekvence 27 MHz

E [392.97...] ... výpočet kapacitní reaktance $X_C = 392.97...$ ohmů

1 **0** **EE** **6** **+/-** **B** ... zadání indukčnosti 10 uH

D [1.6964...+3] ... výpočet indukční reaktance $X_L = 1.6964... \text{ kohmů}$

C [3.4746...-12] ... pro rezonanci na $f=27 \text{ MHz}$ s $L=10 \text{ uH}$ je potřeba
kondenzátor $C = 3.4746... \text{ pF}$

E [1.6964...+3] ... kontrolní výpočet kapacitní reaktance

$X_C = 1.6964... \text{ kohmů}$. Podmínkou rezonance je shoda $X_C = X_L$

ML-30 (EE-12) Konverze sériové/paralelní impedance

->Rs ->Xs ->Rp -> Xp
Rs Xs Rp Xp

Program ML-30 slouží k přepočtu sériových a paralelních impedancí. Po zadání sériové rezistance a reaktance (R_s a X_s spojeny sériově) lze vypočítat náhradní paralelní rezistanci a reaktanci (R_p a X_p spojeny paralelně). Stejně platí i pro opačný převod. Všechny údaje jsou v ohmech.

Lbl **A** Zadání sériové rezistance R_s .

Lbl **B** Zadání sériové reaktance X_s

Lbl **C** Zadání paralelní rezistance R_p

Lbl **D** Zadání paralelní reaktance X_p

Lbl **A'** Výpočet sériové rezistance R_s

Lbl **B'** Výpočet sériové reaktance X_s

Lbl **C'** Výpočet paralelní rezistance R_p

Lbl **D'** Výpočet paralelní reaktance X_p

Example:

RX impedanční metr naměřil při 125 MHz paralelní odpor 75 ohmů a kapacitu 25 pF. Potřebujeme vypočítat náhradní sériové zapojení.

Pgm **29** ... aktivace knihovního programu ML-29

Eng ... technický exponent

1 **2** **5** **EE** **6** **A** ... zadání frekvence $f = 125$ MHz

2 **5** **EE** **1** **2** **+/-** **C** ... zadání capacity $C = 25$ pF

E [50.929...] ... výpočet reaktance $X_C = 50.929...$ ohmů

Pgm **30** ... aktivace knihovního programu ML-30

D ... zadání vypočtené XC jako paralelní reaktance X_p

7 **5** **C** ... zadání paralelní rezistance $R_p = 75$ ohmů

A' [23.856...] ... výpočet sériové rezistance $R_s = 23.856...$ ohmů

B' [34.856...] ... výpočet sériové reaktance $X_s = 34.856...$ ohmů

Pgm **29** ... aktivace knihovního programu ML-29

x<>t ... úschova vypočtené X_s

1 **2** **5** **EE** **6** **A** ... zadání frekvence $f = 125$ MHz

x<>t **E'** [36.528...-12] ... přepočet XC na kapacitu $C = 36.528...$ pF

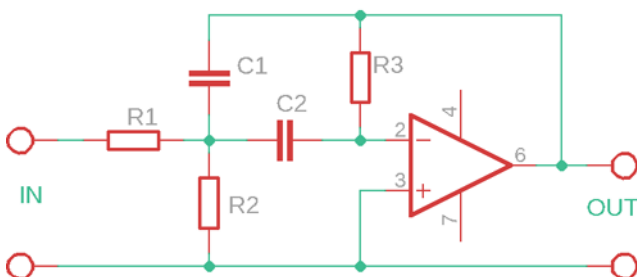
Náhradní sériové zapojení pro 125 MHz má rezistanci 23.856... ohmů a kapacitu 36.528... pF.

ML-31 (EE-13) Aktivní filtr

C1	C2	->BP	->LP	->HP
alpha	A	F	B	

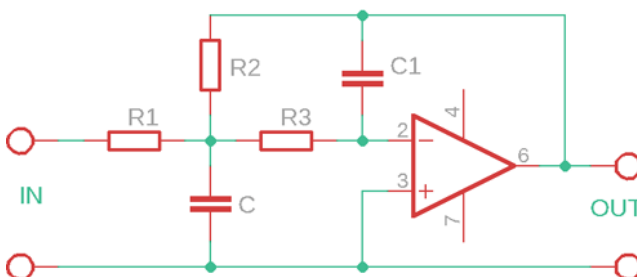
Program ML-31 slouží k výpočtu aktivních filtrů s operačním zesilovačem - pásmová propust BP, dolní propust LP a horní propust HP.

Aktivní pásmová propust (Bandpass BP)



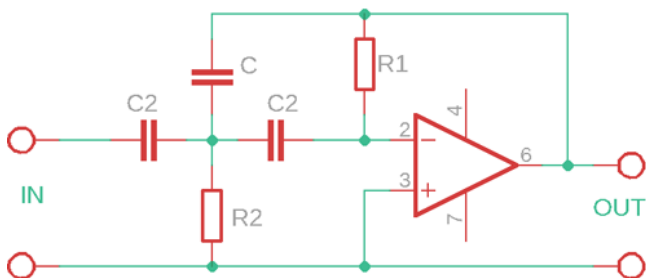
Znamé údaje: Šířka 3-dB pásma 'B' v Hz, zisk ve středu pásma 'A' v dB, frekvence středu pásma 'F' v Hz, kapacity C1 a C2 ve faradech. Program spočítá hodnoty R1, R2 a R3.

Aktivní dolní propust (Lowpass LP)



Znamé údaje: Peaking faktor 'alpha', zisk 'A' v dB, hraniční frekvence 'F' v Hz, C1 ve faradech. Program spočítá C, R1, R2 a R3.

Aktivní horní propust (Highpass HP)



Znamé údaje: Peaking faktor 'alpha', zisk 'A' v dB, hraniční frekvence 'F' v Hz, C2 ve faradech. Program spočítá C, R1 a R2.

Lb| A Zadání peaking faktor 'alpha' (LP, HP)

Peaking faktor udává zaoblení zlomové hrany filtru. Faktor $\alpha = 1$ představuje standardní průběh. Pro $\alpha > 1$ se hrana zlomu zaobljuje. Pro $\alpha < 1$ se na hraně zlomu vytváří ostrá 'špička', frekvence v bodě zlomu křivky budou zdůrazněny.

Lb| B Zadání zisku 'A' v dB (BP, LP, HP)

Lb| C Zadání frekvence 'F' v Hz (BP, LP, HP)

Lb| D Zadání šířky 3-dB pásma 'B' v Hz (BP)

Lb| A' Zadání kapacity C1 ve faradech (BP, LP)

Lb| B' Zadání kapacity C2 ve faradech (BP, HP)

Lb| C' Výpočet pásmové propusti

Před výpočtem zadejte 'B', 'A', 'F', 'C1' a 'C2'. Po stisku C' program zobrazí hodnoty 'R1', 'R2' a 'R3'. Pokračování stiskem **R/S**.

Lb| D' Výpočet dolní propusti

Před výpočtem zadejte 'alpha', 'A', 'F' a 'C1'. Po stisku D' program zobrazí

hodnoty 'C', 'R1', 'R2' a 'R3'. Pokračování stiskem **R/S**.

Lbl E Výpočet horní propusti

Před výpočtem zadejte 'alpha', 'A', 'F' a 'C2'. Po stisku E program zobrazí C, R1 a R2. Pokračování stiskem **R/S**.

Example BP:

Pgm 31 ... aktivace knihovního programu ML-31

Eng ... technický exponent

1 6 D ... zadání 3-dB šířky pásma B = 16 Hz

3 0 B ... zadání zesílení A = 30 dB

1 5 0 C ... zadání středové frekvence F = 150 Hz

1 0 0 EE 9 +/- A B ... zadání C1 a C2 = 100 nF

C [3.145...+3] ... výpočet R1 = 3.145... kohm

R/S [690.0...] ... výpočet R2 = 690.0... ohm

R/S [198.9...+3] ... výpočet R3 = 198.9... kohm

Example LP:

Pgm 31 ... aktivace knihovního programu ML-31

Eng ... technický exponent

1 1 4 1 4 2 A ... zadání peaking faktoru alpha = 1.4142 (=sqrt(2))

2 0 B ... zadání zesílení A = 20 dB

1 EE 3 C ... zadání hraniční frekvence F = 1 kHz

2 0 EE 9 +/- A ... zadání C1 = 20 nF

D [440-9] ... výpočet C = 440 nF

R/S [562.69...] ... výpočet R1 = 562.69... ohm

R/S [5.626...+3] ... výpočet $R_2 = 5.626... \text{ kohm}$

R/S [511.5...] ... výpočet $R_3 = 511.4... \text{ ohm}$

Example HP:

Pgm **31** ... aktivace knihovního programu ML-31

Eng ... technický exponent

.5A ... zadání peaking faktoru $\alpha = 0.5$

6B ... zadání zesílení $A = 6 \text{ dB}$

400C ... zadání hraniční frekvence $F = 400 \text{ Hz}$

47EE9+/-B ... zadání $C_2 = 47 \text{ nF}$

E [23.55-9] ... výpočet $C = 23.55... \text{ nF}$

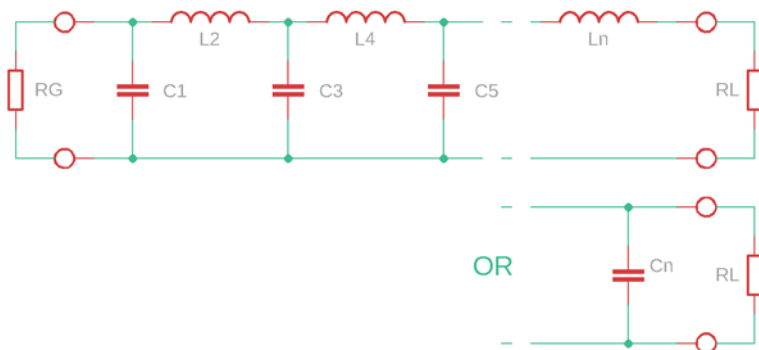
R/S [84.496...+3] ... výpočet $R_1 = 84.496... \text{ kohm}$

R/S [1.692...+3] ... výpočet $R_2 = 1.692... \text{ kohm}$

ML-32 (EE-14) Pasivní dolní propust

n **eps** **R** **fc** **->calc**

Program ML-32 počítá pasivní filtr dolní propusti vyššího řádu, a to buď jako Chebyshevův nebo Butterworthův filtr.



Pasivní dolní propust je tvořena střídajícími se kondenzátory zapojenými paralelně a cívkami zapojenými sériově, s číslováním od 1 do n . Pro sudé číslo ' n ' je filtr zakončen cívkou, pro liché ' n ' je zakončen kondenzátorem. Rezistor R_G představuje interní odpor generátoru signálu. Rezistor R_L je vstupní odpor zátěže. Program předpokládá, že hodnoty obou rezistorů se shodují, $R_G = R_L = R$.

Lbl A Zadání řádu filtru ' n '

Lbl B Zadání zvlnění ϵ_{ps} v dB

Zvlnění ϵ_{ps} je povolené zvlnění průběhu filtru v dolních (propustných) pásmech, platné pro Chebyshevův filtr. V případě Butterworthova filtru se zadává $\epsilon_{ps} = 0$.

Lbl C Zadání zakončovacího odporu R v ohmech

Lbl D Zadání mezní frekvence f_c v Hz

Lbl E Výpočet parametrů filtru

Program průběžně zobrazuje střídavě hodnoty Ck a Lk. Pokračování s **R/S**.

Example 1, Butterworthův filtr 9. řádu:

Pgm 32 ... aktivace knihovního programu ML-32

Eng ... technický exponent

9 A ... Zadání řádu filtru $n = 9$

0 B ... Zadání zvlnění $\text{eps} = 0$, zvolí Butterworthův filtr

2 EE 3 C ... Zadání zakončovacího odporu $R = 2 \text{ kohm}$

1 0 EE 3 D ... Zadání mezní frekvence $f_c = 10 \text{ kHz}$

E [2.763...-9] ... Výpočet $C1 = 2.763... \text{ nF}$

R/S [31.83...-3] ... Výpočet $L2 = 31.83... \text{ mH}$

R/S [12.19...-9] ... Výpočet $C3 = 12.19... \text{ nF}$

R/S [59.82...-3] ... Výpočet $L4 = 59.82... \text{ mH}$

R/S [15.91...-9] ... Výpočet $C5 = 15.91... \text{ nF}$

R/S [59.82...-3] ... Výpočet $L6 = 59.82... \text{ mH}$

R/S [12.19...-9] ... Výpočet $C7 = 12.19... \text{ nF}$

R/S [31.83...-3] ... Výpočet $L8 = 31.83... \text{ mH}$

R/S [2.763...-9] ... Výpočet $C9 = 2.763... \text{ nF}$

Example 2, Chebyshevův filtr 7. řádu:

Pgm 32 ... aktivace knihovního programu ML-32

Eng ... technický exponent

7 A ... Zadání řádu filtru $n = 7$

. 5 B ... Zadání zvlnění $\text{eps} = 0.5 \text{ dB}$, zvolí Chebyshevův filtr

1 EE 3 C ... Zadání zakončovacího odporu $R = 1 \text{ kohm}$

3 . 5 EE 3 D ... Zadání mezní frekvence $f_c = 3.5 \text{ kHz}$

E [78.99...-9] ... Výpočet $C1 = 78.99... \text{ nF}$

R/S [57.21...-3] ... Výpočet $L2 = 57.21... \text{ mH}$

R/S [119.9...-9] ... Výpočet $C3 = 119.9... \text{ nF}$

R/S [61.13...-3] ... Výpočet $L4 = 61.13... \text{ mH}$

R/S [119.9...-9] ... Výpočet $C5 = 119.9... \text{ nF}$

R/S [57.21...-3] ... Výpočet $L6 = 57.21... \text{ mH}$

R/S [78.99...-9] ... Výpočet $C7 = 78.99... \text{ nF}$

ML-33 (EE-15) Konvoluce signálu

n0 dt ->y(t)

Konvoluce je odezva lineárního systému na vstupní signál. V hlavním uživatelském programu je nejdříve potřeba vytvořit funkci označenou **Lbl A'**, která navrací průběh vstupního signálu v závislosti na čase $x(t)$. Dále je potřeba vytvořit funkci označenou **Lbl B'**, která navrací průběh výstupního signálu soustavy v závislosti na čase $h(t)$, jako reakce na vstupní impulzní signál. Obě funkce nesmí používat klávesy **=** ani **CLR**. Program spočítá výstupní signál $y(t)$.

Lbl A Zadání počtu úseků n0 v každém přírůstku času dt

Lbl B Zadání přírůstku času dt

Lbl C Výpočet hodnot $y(t)$

Funkce zobrazí v horním řádku displeje aktuální čas 't' a v dolním řádku hodnotu výstupního signálu $y(t)$. Výpočet probíhá od času $0+dt$, v čase 0 je $y(0) = 0$. Pokračování s **R/S**. Návrat displeje na běžný mód zobrazení s **Op 1D**.

Example:

Vstupní signál má průběh $x(t) = 2^t$ pro $t \leq 0.3$ jinak 0. Odezva na impulz má tvar $h(t) = 10 \cdot \exp(-5 \cdot t)$.

RST LRN ... aktivace programovacího módu

Lbl A' ... návěští začátku funkce vstupního signálu $x(t)$

[CE] x 2 [] x<>t ... hodnota 2^t do registru T

[] 6 x>=1 Nop ... je-li $2^t < 0.6$, skok na návěští Nop

CP ... pro $t > 0.3$ vynulování registru T

Lbl Nop ... návěští pro případ $t \leq 0.3$

x<>t ... návrat registru T na displej

RTN ... konec funkce **A'** (= **INV** **SBR**)

Lbl **B'** ... návěští začátku funkce odezvy $h(t)$

(**(** **+/-** **x** **5** **)** ... hodnota $-5 \cdot t$

INV **lnx** **x** **1** **0** **)** ... hodnota $10 \cdot \exp(-5 \cdot t)$

RTN ... konec funkce **B'** (= **INV** **SBR**)

LRN ... návrat z programovacího módu

Pgm **33** ... aktivace knihovního programu ML-33

4 **A** ... zadání počtu úseků pro rozdělení času dt , $n_0 = 4$

1 **B** ... zadání časového přírůstku $dt = 0.1$

C [0.0861...] ... výpočet $y(0.1) = 0.0861...$

R/S [0.2960...] ... výpočet $y(0.2) = 0.2960...$

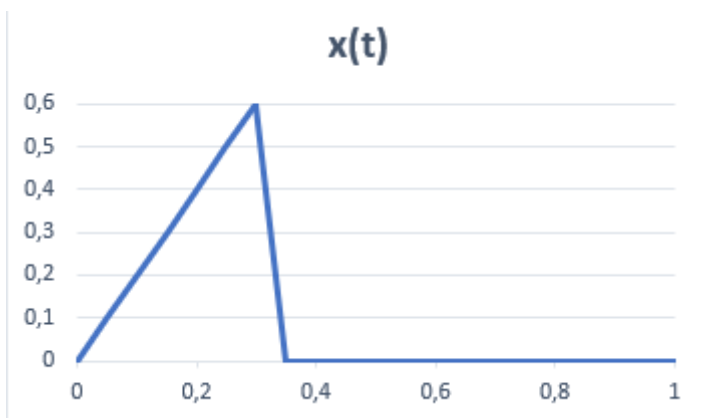
R/S [0.5808...] ... výpočet $y(0.3) = 0.5808...$

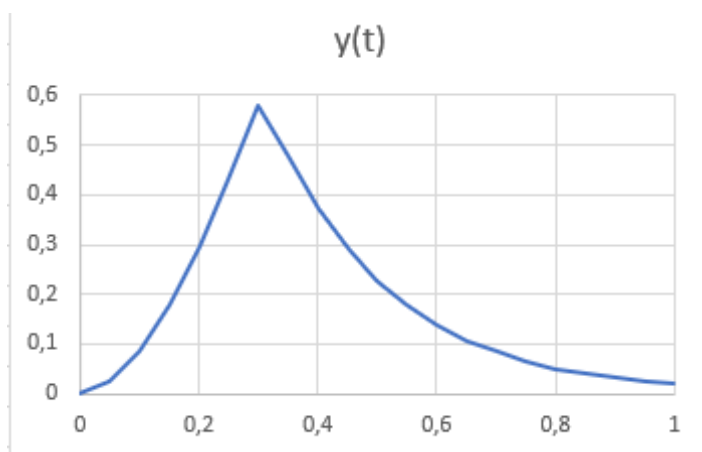
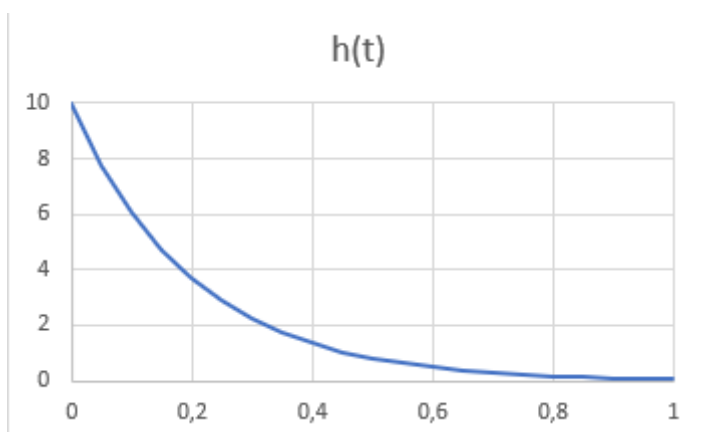
R/S [0.3978...] ... výpočet $y(0.4) = 0.3978...$

R/S [0.2412...] ... výpočet $y(0.5) = 0.2412...$

R/S [0.1463...] ... výpočet $y(0.6) = 0.1463...$

R/S [0.0887...] ... výpočet $y(0.7) = 0.0887...$





ML-34 (EE-17) Diskrétní Fourierova transformace

N **n,f(n)** ->DFT **n,F(n)** ->IDFT

Program ML-34 počítá diskrétní Fourierovu transformaci. Převádí vzorky signálu reálného času na frekvenční spektrum (DFT, diskrétní Fourierova transformace) a též zpětný převod ze spektra na časový průběh (IDFT, inverzní diskrétní Fourierova transformace). Program podporuje až 100 vzorků při DFT převodu a 50 vzorků při IDFT převodu.

Lbl **A** Zadání počtu vzorků N

Pro DFT max. 100 vzorků, pro IDFT max. 50 vzorků (zadávají se páry hodnot, amplituda a fáze).

Lbl **B** Zadání časových vzorků f(n)

Zadejte index první položky n (= 0...N-1), stiskněte B a zadávejte časová data f(n). Pokračování s **R/S**. Časové vzorky jsou v paměti uloženy v datových registrech R00 až R99.

Lbl **C** Výpočet DFT

Stiskem **C** se časové vzorky přepočtou na frekvenční vzorky. Frekvenční vzorky se zobrazují po dvojicích hodnot - zobrazí se amplituda amp(n), stiskněte **R/S**, zobrazí se fáze phase(n), stiskněte **R/S**.

Lbl **D** Zadání frekvenčních vzorků F(n)

Zadejte index první položky n (= 0...N-1), stiskněte **D** a zadávejte dvojice vzorků - zadejte amplitudu amp(n), stiskněte R/S, zadejte fázi phase(n), stiskněte **R/S**. Frekvenční vzorky jsou v paměti uloženy převedené na reálnou a imaginární část čísla. V registrech R00 až R49 jsou uloženy reálné části čísel, v registrech R50 až R99 jsou uloženy imaginární části čísel.

Lbl **E** Výpočet IDFT

Stiskem **E** se frekvenční vzorky přepočtou na časové vzorky. Pokračování s **R/S**.

Example:

Frekvenční spektrum obsahuje 32 vzorků (= N). Vzorek n = 2 má hodnotu (0 - 16i), vzorek n = 30 má hodnotu (0 + 16i), ostatní vzorky mají hodnotu 0. Vzorky budou zapsány přímo do paměti ve formátu komplexních čísel. Zpětnou transformaci IDFT bychom měli získat původní signál, který měl tvar $S = \sin(n \cdot \pi/8)$, kde $n = 0 \dots N-1$. Signálem je sinusovka o frekvenci 1/8.

Pgm **34** ... aktivace knihovního programu ML-34

CMS ... vynulování všech datových registrů

3 **2** **A** ... zadání počtu vzorků $N = 32$

1 **6** **STO** **80** ... vložení imaginární složky vzorku (0 + 16i) s indexem 30

+/- **STO** **52** ... vložení imaginární složky vzorku (0 - 16i) s indexem 2

Fix **4** ... vzorky budeme číst na 4 desetinná místa

E ... provedení převodu na časové vzorky

[0] ... $f(0) = 0$

R/S [0.3827] ... $f(1) = 0.3827$, očekáváme $\sin(1 \cdot \pi/8) = 0.3827$

R/S [0.7071] ... $f(2) = 0.7071$, očekáváme $\sin(2 \cdot \pi/8) = 0.7071$

R/S [0.9239] ... $f(3) = 0.9239$, očekáváme $\sin(3 \cdot \pi/8) = 0.9239$

R/S [1] ... $f(4) = 1$, očekáváme $\sin(4 \cdot \pi/8) = 1$

R/S [0.9239] ... $f(5) = 0.9239$, očekáváme $\sin(5 \cdot \pi/8) = 0.9239$

R/S [0.7071] ... $f(6) = 0.7071$, očekáváme $\sin(6 \cdot \pi/8) = 0.7071$

R/S [0.3827] ... $f(7) = 0.3827$, očekáváme $\sin(7 \cdot \pi/8) = 0.3827$

R/S [0] ... $f(8) = 0$, očekáváme $\sin(8 \cdot \pi/8) = 0$

R/S [-0.3827] ... $f(9) = -0.3827$, očekáváme $\sin(9 \cdot \pi/8) = -0.3827$

R/S [-0.7071] ... $f(10) = -0.7071$, očekáváme $\sin(10 \cdot \pi/8) = -0.7071$

R/S [-0.9239] ... $f(11) = -0.9239$, očekáváme $\sin(11 \cdot \pi/8) = -0.9239$

R/S [-1] ... $f(12) = -1$, očekáváme $\sin(12 \cdot \pi/8) = -1$

R/S [-0.9239] ... $f(13) = -0.9239$, očekáváme $\sin(13 \cdot \pi/8) = -0.9239$

R/S [-0.7071] ... $f(14) = -0.7071$, očekáváme $\sin(14 \cdot \pi/8) = -0.7071$

R/S [-0.3827] ... $f(15) = -0.3827$, očekáváme $\sin(15 \cdot \pi/8) = -0.3827$

... dále se pro $f(16)$ až $f(31)$ opakují vzorky jako pro $f(0)$ až $f(15)$.

ML-35 Ohmův zákon

->U ->I ->R ->Pui Pu->R
U I R Pr->U Pr->I

Program ML-35 slouží k výpočtům napětí, proudu a odporu podle Ohmova zákona. Po zadání dvou z veličin se vypočte třetí veličina. Doplnkově se počítá ztrátový výkon na rezistoru.

Lbl A Zadání napětí U

Lbl A' Výpočet napětí U (z proudu I a odporu R)

Lbl B Zadání proudu I

Lbl B' Výpočet proudu I (z napětí U a odporu R)

Lbl C Zadání odporu R

Lbl C' Výpočet odporu R (z napětí U a proudu I)

Lbl D Zadání ztrátového výkonu P a výpočet napětí U na odporu R

Lbl D' Výpočet ztrátového výkonu P (z napětí U a proudu I)

Lbl E Zadání ztrátového výkonu P a výpočet proudu I odporem R

Lbl E' Zadání ztrátového výkonu P a výpočet odporu R s napětím U

Example

Pgm 35 ... aktivace knihovního programu ML-35

Eng ... technický exponent

5 A ... zadání napětí U = 5V

4 EE 3 +/- B ... zadání proudu I = 4 mA

C' [1.25+3] ... výsledný odpor R = 1.25 kohmů

D' [20-3] ... ztrátový výkon na odporu je $P = 20 \text{ mW}$

1 E [25] ... pro ztrátový výkon 1 W by byl odpor $R = 25 \text{ ohmů}$

C ... vypočtený odpor $R = 25 \text{ ohmů}$ se použije na vstupu

B' [200-3] ... odporem 25 ohmů by při 5 V tekla proud 200 mA

ML-36 Sérové a paralelní řazení

RLC +RLs,Cp +RLp,Cs

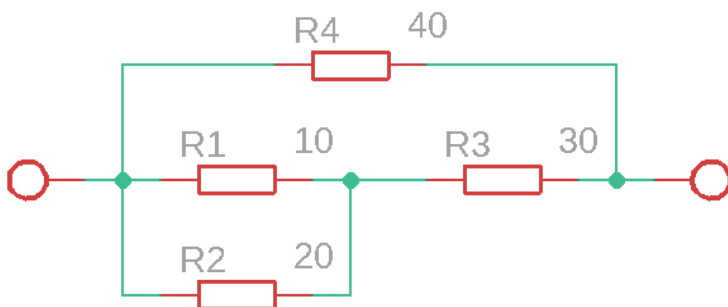
Program ML-36 počítá výslednou hodnotu odporu, indukčnosti nebo kapacity, spojené sériově či paralelně.

Lbl A Zadání hodnoty prvního odporu, kapacity, indukčnosti

Lbl B Přidání sériového odporu nebo indukčnosti, paralelní kapacity

Lbl C Přidání paralelního odporu nebo indukčnosti, sériové kapacity

Example:



Pgm 36 ... aktivace knihovního programu ML-36

1 0 A ... zadání prvního rezistoru R1 = 10 ohmů

2 0 C [6.666...] ... přidání paralelního R2 = 20 ohmů

3 0 B [36.666...] ... přidání sériového R3 = 30 ohmů

4 0 C [19.13...] ... přidání paralelního R4 = 40 ohmů

... Výsledný odpor je 19.13... ohmů.

ML-37 (AV-23) Konverze časové zóny

->date'
zone date time dt ->time'

Program ML-37 počítá změnu data a času při přeletu do jiného časového pásma. Datum je ve formátu MM.DD (měsíc, den), čas ve formátu HH.MMSS (hodiny, minuty, sekundy).

Lbl A Zadání časové zóny (západ: -, východ +)

Časová zóna představuje odchylku času od UTC času. Směrem na západ od nultého poledníku v Greenwich má zóna záporné znaménko, směrem na východ má kladné znaménko.

Note: Původní program AV-23 používá časové zóny pro letectví, kdy zóna představuje korekci pro přepočítání lokálního času na UTC čas. Tedy časové zóny mají opačné znaménko.

Lbl B Zadání výchozího data ve formátu MM.DD

Lbl C Zadání výchozího času ve formátu HH.MMSS

Lbl D Zadání přírůstku času 'dt' ve formátu HH.MMSS

Maximální povolená hodnota 'dt' je +- 648 hodin (tj. 27 dnů).

Lbl E Výpočet cílového času ve formátu HH.MMSS

Lbl E Výpočet cílového data ve formátu MM.DD

Program nezohledňuje přechodný rok, únor počítá s 28 dny. Při přeletu přes datum 29. února v přechodný rok je nutné výsledné datum opravit o 1 den.

Example 1:

Let z Tusconu (zóna -7) do New Yorku (zóna -5) zabere 5 hodin, 22 minut. Start je 31. prosince v 22:10.

Pgm **37** ... aktivace knihovního programu ML-37

7 **+/-** **A** ... startovací zóna = -7

1 **2** **3** **1** **B** ... startovací datum = 31. prosince

2 **2** **1** **0** **C** ... startovací čas = 22:10

5 **2** **2** **D** ... doba letu = 5 hodin, 22 minut

5 **+/-** **A** ... cílová zóna = -5

E [5.32] ... výpočet času příletu = 5:32:00

E' [1.01] ... výpočet data příletu = 1. ledna

Example 2:

Přejete si letět z Honolulu, Hawaii (zóna -10) do New Dehli, Indie (zóna +5.5). Doba letu bude s mezipřistáním 35 hodin, 27 minut. Odlet bude 19. září v 8:40.

Pgm **37** ... aktivace knihovního programu ML-37

1 **0** **+/-** **A** ... startovací zóna = -10

9 **1** **9** **B** ... startovací datum = 19. září

8 **4** **0** **C** ... startovací čas = 8:40

3 **5** **2** **7** **D** ... doba letu = 35 hodin, 27 minut

5 **5** **A** ... cílová zóna = +5.5

E [11.37] ... výpočet času příletu = 11:37:00

E' [9.21] ... výpočet data příletu = 21. září

Váš kontakt v New Dehli se opozdí o 6 dnů. Setkáte se na letišti 27. září v 15:00. Pokud doba letu zůstává stejná, kdy musíte odletět z Honolulu?

5 **5** **A** ... cílová zóna = +5.5

9 **2** **7** **B** ... cílové datum = 27. září

1 **5** **C** ... cílový čas = 15:00

3 5 . 2 7 +/- D ... doba letu = - 35 hodin, 27 minut (posun času zpět)

1 0 +/- A ... startovací zóna = -10

E [12.03] ... výpočet času odletu = 12:03:00

E' [9.25] ... výpočet data odletu = 25. září

Example 3:

Je-li v Chicagu (zóna -6) čas 21.15 a datum 23. listopadu, jaký je datum a čas v Praze (zóna 1)?

Pgm 37 ... aktivace knihovního programu ML-37

6 +/- A ... první zóna = -6

1 1 . 2 3 B ... první datum = 23. listopadu

2 1 . 1 5 C ... první čas = 21:15

0 D ... není posun času, zajímá nás aktuální čas

1 A ... druhá zóna = +1

E [4.1500] ... druhý čas = 4:15:00

E' [11.24] ... druhé datum = 24. listopadu

ML-38 (MU-06) Shellovo třídění

Geom Mean

N Enter Sort View Median

Program ML-38 seřídí zadaná data metodou 'Shell sort' a zjistí median dat (tj. střední hodnotu).

Lbl A Zadání počtu prvků N (max. 100)

Lbl B Zadání dat od zadaného indexu n (0...N-1)

Pokračování stiskem **R/S**.

Lbl C Setřídění dat

Lbl D Zobrazení dat od zadaného indexu n (0...N-1)

Pokračování stiskem **R/S**.

Lbl D' Výpočet geometrického průměru

Lbl E Zobrazení medianu (střední hodnota dat)

Lbl E' Výpočet aritmetického průměru

Example:

Pgm 38 ... aktivace knihovního programu ML-38

1 0 A ... zadání počtu prvků N = 10

0 B ... start zadávání dat od indexu 0

1 0 . 6 R/S ... zadání dat d0 = 10.6

5 1 1 2 R/S ... zadání dat d1 = 5.12

1 1 R/S ... zadání dat d2 = 11

9 **|** **2** **R/S** ... zadání dat d3 = 9.2

4 **|** **3** **+/-** **R/S** ... zadání dat d4 = -4.3

1 **|** **4** **5** **+/-** **R/S** ... zadání dat d5 = -1.45

| **4** **R/S** ... zadání dat d6 = 0.4

3 **7** **R/S** ... zadání dat d7 = 37

| **1** **R/S** ... zadání dat d8 = 0.1

8 **|** **3** **R/S** ... zadání dat d9 = 8.3

C ... setřídění dat

0 **D** ... zobrazení dat od indexu 0

[-4.3] ... zobrazení dat d0 = -4.3

R/S **[-1.45]** ... zobrazení dat d1 = -1.45

R/S **[0.1]** ... zobrazení dat d2 = 0.1

R/S **[0.4]** ... zobrazení dat d3 = 0.4

R/S **[5.12]** ... zobrazení dat d4 = 5.12

R/S **[8.3]** ... zobrazení dat d5 = 8.3

R/S **[9.2]** ... zobrazení dat d6 = 9.2

R/S **[10.6]** ... zobrazení dat d7 = 10.6

R/S **[11]** ... zobrazení dat d8 = 11

R/S **[37]** ... zobrazení dat d9 = 37

E **[8.3]** ... zobrazení mediánu = 8.3

E' **[7.597]** ... zobrazení aritmetického průměru = 7.597

D' **[3.6508...]** ... zobrazení geometrického průměru = 3.6508...

ML-39 (MU-09) Rozklad na prvočinitele

Prime

Program ML-39 rozloží celé číslo na prvočinitele.

Lbl **A** Zadání celého čísla a vyhledání prvního prvočinitele

Pokračování stiskem **R/S**, 1 = není další prvočinitel.

Example:

Pgm **39** ... aktivace knihovního programu ML-39

9 **8** **7** **6** **5** **4** **3** **2** **1** **A** ... zadání čísla k rozkladu

[3] ... zobrazení 1. prvočinitele

R/S **[3]** ... zobrazení 2. prvočinitele

R/S **[17]** ... zobrazení 3. prvočinitele

R/S **[17]** ... zobrazení 4. prvočinitele

R/S **[379721]** ... zobrazení 5. prvočinitele

R/S **[1]** ... není další prvočinitel

Rozklad čísla 987654321 = $3 \cdot 3 \cdot 17 \cdot 17 \cdot 379721$

ML-40 (MU-21) Aritmetika s proměnnými

->A ->B ->C ->D ->E
A B C D E

Program ML-40 umožňuje snadné výpočty s proměnnými.

[Lb] [A] až [E] Vyvolání proměnné A až E

[Lb] [A'] až [E'] Nastavení hodnoty proměnné A až E

Example:

Výpočet akceleraace $a = 2 * d / t^2$ (d = vzdálenost, t = čas) a rychlosti $v = a * t$.

[Pgm] [40] ... aktivace knihovního programu ML-40

[2] [5] [A'] ... uložení distance $d = 25$ do proměnné A

[1] [.] [7] [B'] ... uložení času $t = 1.7$ do proměnné B

[2] [x] [A] [:] [B] [x^2] [=] [17.30...] ... výpočet akceleraace $a = 2 * d / t^2$

[x] [B] [=] [29.41...] ... výpočet rychlosti $v = a * t$

[1] [.] [5] [B'] ... uložení času $t = 1.5$ do proměnné B

[2] [x] [A] [:] [B] [x^2] [=] [22.22...] ... výpočet akceleraace $a = 2 * d / t^2$

[x] [B] [=] [33.33...] ... výpočet rychlosti $v = a * t$

[1] [.] [3] [B'] ... uložení času $t = 1.3$ do proměnné B

[2] [x] [A] [:] [B] [x^2] [=] [29.58...] ... výpočet akceleraace $a = 2 * d / t^2$

[x] [B] [=] [38.46...] ... výpočet rychlosti $v = a * t$

ML-41 (MU-14) Interpolace

N Enter $x \rightarrow f(x)$

Program ML-14 interpoluje zadaná data polynomem (N-1)tého řádu, použitím Aitkenovy metody.

Lbl A Zadání počtu vzorků dat N (max.33)

Lbl B Zadání párů (x,y) od zadaného indexu n (0...N-1)

Při zadávání se zadá hodnota x_i , stiskne **R/S**, zadá hodnota y_i a stiskne opět **R/S**.

Lbl C Výpočet interpolované hodnoty $x \rightarrow f(x)$

Výpočet další hodnoty stiskem **C** nebo **R/S**.

Example:

Pgm 41 ... aktivace knihovního programu ML-41

5 A ... zadání počtu vzorků dat N = 5

0 B ... zahájení zadávání dat od indexu n = 0

6 R/S **2 2 5 7 R/S** ... zadání vzorku (x_0, y_0) = (0.6, 0.2257)

7 R/S **2 5 8 0 R/S** ... zadání vzorku (x_1, y_1) = (0.7, 0.2580)

9 R/S **3 1 5 9 R/S** ... zadání vzorku (x_2, y_2) = (0.9, 0.3159)

1 R/S **3 4 1 3 R/S** ... zadání vzorku (x_3, y_3) = (1, 0.3413)

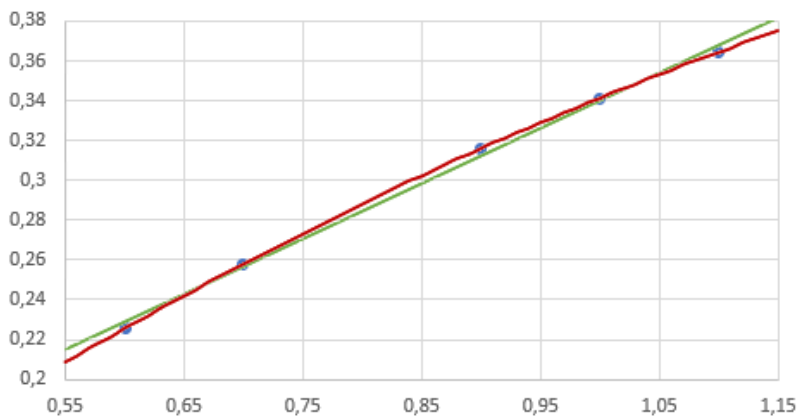
1 1 R/S **3 6 4 3 R/S** ... zadání vzorku (x_4, y_4) = (1.1, 0.3643)

8 C [0.28811] ... interpolace $f(0.8) = 0.28811$

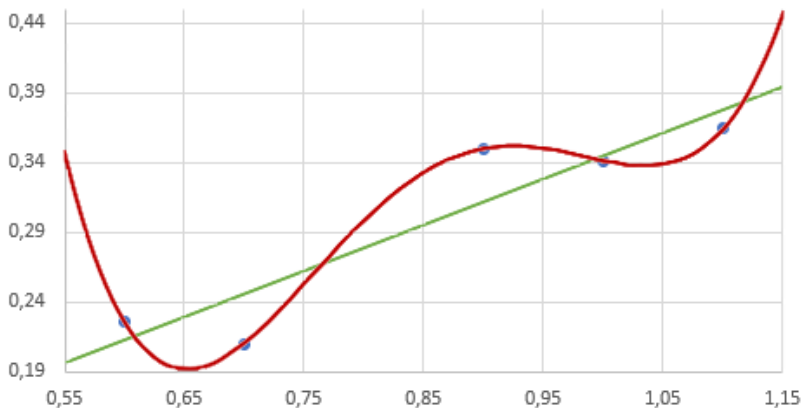
Note: Interpolace se od aproximace liší tím, že při aproximaci jsou body proloženy křivkou snažící se napodobit průběh dat rovnoměrnou hladkou křivkou, naproti tomu interpolace striktně dodržuje průchod křivky datovými

body, i za cenu zvlněného průběhu křivky.

Graf interpolace bodů podle příkladu. Interpolační křivka (červenou barvou) má zde velmi dobrý průběh. Pro porovnání aproximace lineární regresí (zelenou barvou).



Změníme-li druhý a třetí bod z příkladu na hodnoty $y=0,21$ a $0,35$, interpolační křivka (červenou barvou) se snaží i nadále udržet průchod křivky danými body a dostává značně zvlněný tvar. Pro porovnání opět aproximace lineární regresí (zelenou barvou).



ML-42 (MU-16) Vyhledání minima a maxima

Max x->crit next f(x) f'(x)

Program ML-42 vyhledává minima a maxima funkce. Před použitím je nutné v hlavním programu vytvořit funkci označenou **Lbl A**, která přepočítá x na f(x). Funkce nesmí používat **=** ani **CLR**.

Program při hledání kritických bodů rozdělí interval od výchozí hodnoty 'x' k maximální hodnotě 'x' na 100 dílů, ve kterých provádí hledání. Změní-li derivace funkce na začátku a konce dílu znaménko, jedná se o kritický bod a program vyhledá přesné místo metodou půlení intervalu.

Lbl A **Zadání maximální hodnoty 'x' k hledání kritických bodů**

Lbl B **Vyhledání kritického bodu 'x' od zadaného počátečního 'x'**

Je-li kritický bod nalezen, obsahuje registr T typ bodu: -1 maximum, +1 minimum. Není-li kritický bod nalezen, bliká na displeji počáteční hodnota 'x'. Pokračujte v hledání dalšího bodu stiskem **C** nebo **R/S**.

Lbl C **Vyhledání dalšího kritického bodu**

Je-li další kritický bod nalezen, obsahuje registr T typ bodu: -1 maximum, +1 minimum. Není-li kritický bod nalezen, bliká na displeji poslední nalezený kritický bod 'x'. Pokračujte v hledání dalšího bodu stiskem **C** nebo **R/S**.

Lbl D **Výpočet hodnoty funkce pro zadané 'x'**

Lbl E **Výpočet derivace funkce pro zadané 'x'**

Před výpočtem derivace je nutné zadat maximum 'x' pomocí **A** a minimum 'x' pomocí **B**, ze kterých si program připraví epsilon 'x' pro test derivace.

Example, $f(x) = x^3 - x^2 - x + 2$:

RST LRN ... aktivace programovacího módu

Lbl **A'** ... návěští začátku funkce

(**STO** **10** **x^2** **x** **RCL** **10** ... x^3

- **RCL** **10** **x^2** ... $-x^2$

- **RCL** **10** **+** **2** **)** ... $-x + 2$

RTN ... konec funkce A' (= **INV** **SBR**)

LRN ... konec programovacího módu

Pgm **42** ... aktivace knihovního programu ML-42

2 **A** ... zadání maxima k hledání kritického bodu = 2

1 **+/-** **B** [-0.33333...] ... vyhledání prvního kritického bodu $x_1 = -0.33333...$

D [2.1851...] ... hodnota funkce v kritickém bodě $f(x) = 2.1851...$

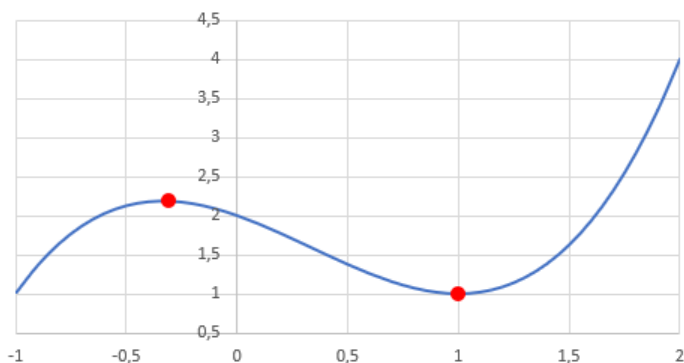
x<>t [-1] ... kritický bod je maximum

C [1] ... vyhledání dalšího kritického bodu $x_2 = 1$

D [1] ... hodnota funkce v kritickém bodě $f(x) = 1$

x<>t [1] ... kritický bod je minimum

C [bliká 1] ... není další kritický bod



ML-43 Kontrolní součet

CCITT CCITTB Dallas XOR INIT
CRC32 IBM Modbus Kermit XModem

Program ML-43 počítá kontrolní součet zadaných dat různými metodami. Před výpočtem stiskněte nejdříve klávesu **E**, která provede přípravu pro počítání nové řady dat a přepne kalkulačtor do HEX módu. Poté zadávejte bajty dat (v HEX kódu) a stiskem příslušné metody vypočítáte následující hodnotu kontrolního součtu. Pokračování dalším bajtem po stisku **R/S**.

Lb **A** Výpočet CRC-32 (32 bitů)

Zadání bajtu v HEX kódu (00...FF) a výpočet následující hodnoty metodou CRC-32 (32 bitů). Pokračování dalším bajtem s **R/S** nebo **A**.

Použitý CRC-32-IEEE802.3 počítá pomocí polynomu $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ s inicializací 0xFFFFFFFF a používá se v Ethernetu a MPEG2.

Kontrolní vzorky:

"123456789" = 31 32 33 34 35 36 37 38 39 -> CBF43926

FC 05 4A -> A8E10F6D

Lb **B** Výpočet CRC-IBM (16 bitů)

Zadání bajtu v HEX kódu (00...FF) a výpočet následující hodnoty metodou CRC-IBM (16 bitů). Pokračování dalším bajtem s **R/S** nebo **B**.

Použitý CRC-IBM počítá s polynomem $x^{16} + x^{15} + x^2 + 1$ s inicializací 0 a používá se v radičích disků a na sběrnici Dallas Maxim 1-Wire.

Kontrolní vzorky:

"123456789" = 31 32 33 34 35 36 37 38 39 -> BB3D

FC 05 4A -> 9742

Lbl C Výpočet CRC-Modbus (16 bitů)

Zadání bajtu v HEX kódu (00...FF) a výpočet následující hodnoty metodou CRC-Modbus (16 bitů). Pokračování dalším bajtem s **R/S** nebo **C**.

Použitý CRC-Modbus počítá s polynomem $x^{16} + x^{15} + x^2 + 1$ s inicializací 0xFFFF a používá se v komunikačním protokolu Modbus.

Kontrolní vzorky:

"123456789" = 31 32 33 34 35 36 37 38 39 -> 4B37

FC 05 4A -> 5733

Lbl D Výpočet CRC-Kermit (16 bitů)

Zadání bajtu v HEX kódu (00...FF) a výpočet následující hodnoty metodou CRC-Kermit (16 bitů). Pokračování dalším bajtem s **R/S** nebo **D**.

Použitý CRC-Kermit (původní CRC-CCITT) počítá s polynomem $x^{16} + x^{12} + x^5 + 1$ s inicializací 0 a používá se v komunikačním protokolu Kermit.

Kontrolní vzorky:

"123456789" = 31 32 33 34 35 36 37 38 39 -> 8921

FC 05 4A -> 71BA

Lbl E Výpočet CRC-XModem (16 bitů)

Zadání bajtu v HEX kódu (00...FF) a výpočet následující hodnoty metodou CRC-XModem (16 bitů). Pokračování dalším bajtem s **R/S** nebo **E**.

Použitý CRC-XModem počítá s polynomem $x^{16} + x^{12} + x^5 + 1$ s inicializací 0 a používá se v komunikačním protokolu XModem. Tuto metodu CRC používá také kalkulátor ET-58 k vnitřní kontrole integrity ROM paměti, protože pro ni existuje rychlá nenáročná metoda výpočtu, a tak je vhodná pro použití v malých zařízeních.

Kontrolní vzorky:

"123456789" = 31 32 33 34 35 36 37 38 39 -> 31C3

FC 05 4A -> 8048

Lb| A' Výpočet CRC-CCITT (16 bitů)

Zadání bajtu v HEX kódu (00...FF) a výpočet následující hodnoty metodou CRC-CCITT (16 bitů). Pokračování dalším bajtem s **R/S** nebo **A'**.

Použitý CRC-CCITT počítá s polynomem $x^{16} + x^{12} + x^5 + 1$ s inicializací 0xFFFF.

Kontrolní vzorky:

"123456789" = 31 32 33 34 35 36 37 38 39 -> 29B1

FC 05 4A -> 4CD4

Lb| B' Výpočet CRC-CCITT-B (16 bitů)

Zadání bajtu v HEX kódu (00...FF) a výpočet následující hodnoty metodou CRC-CCITT-B (16 bitů). Pokračování dalším bajtem s **R/S** nebo **B'**.

Použitý CRC-CCITT-B počítá s polynomem $x^{16} + x^{12} + x^5 + 1$ s inicializací 0x1D0F.

Kontrolní vzorky:

"123456789" = 31 32 33 34 35 36 37 38 39 -> E5CC

FC 05 4A -> 9144

Lb| C' Výpočet CRC-Dallas (8 bitů)

Zadání bajtu v HEX kódu (00...FF) a výpočet následující hodnoty metodou CRC-Dallas (8 bitů). Pokračování dalším bajtem s **R/S** nebo **C'**.

Použitý CRC-Dallas počítá s polynomem $x^8 + x^5 + x^4 + 1$ s inicializací 0

a používá se na sběrnici Dallas Maxim 1-Wire.

Kontrolní vzorky:

"123456789" = 31 32 33 34 35 36 37 38 39 -> A1

FC 05 4A -> F1

Lbl D' Výpočet CRC-XOR (16 bitů)

Zadání bajtu v HEX kódu (00...FF) a výpočet následující hodnoty metodou CRC-XOR (16 bitů). Pokračování dalším bajtem s **R/S** nebo **C'**.

Použitý CRC-XOR počítá kontrolní součet s XOR operací s rotací vlevo, s inicializací 0. Používá se v malých zařízeních ke kontrole integrity paměti ROM, kvůli nenáročnému použití. Není-li k dispozici instrukce pro bitovou rotaci, lze nahradit instrukcí ADD a následným přičtením carry přenosu.

Kontrolní vzorky:

"123456789" = 31 32 33 34 35 36 37 38 39 -> 406A

FC 05 4A -> 0760

Example:

Pgm 43 ... aktivace knihovního programu ML-43

E' ... inicializace nové posloupnosti CRC

1 2 A ... zadání 1. bajtu 12 (v HEX kódu) s výpočtem CRC-32

3 4 R/S 5 6 R/S ... zadání 2. a 3. bajtu 34, 56

[D9C1A93A] ... výsledný CRC-32 = D9C1A93A (HEX)

ML-44 (LE-09) Hra Codebreaker

Guess

Start

Při hře Codebreaker hádáte číslo o 4 číslicích 1...9 (bez vícenásobného výskytu číslic), kalkulátor zobrazuje počet uhádnutých číslic.

Lbl **A** Zadání odhadu čísla o 4 číslicích 1...9

Kalkulátor vrátí výsledek odhadu jako číslo N.R, kde N představuje počet číslic na správných pozicích a R je počet číslic na nesprávných pozicích. Zobrazí-li výsledek 4.0, číslo bylo správně uhádnuto.

Lbl **E** Start nové hry

*Note: Pro reprodukovatelnost náhody je možné inicializovat generátor náhody operací **Op** **52**. Běžně není potřeba generátor náhody inicializovat.*

Example:

Pgm **44** ... aktivace knihovního programu ML-44

2 **0** **7** **Op** **52** ... inicializace generátoru náhody - pouze pro demonstraci

1 **2** **3** **4** **A** [1.1] ... 1 číslice správně, 1 číslice na nesprávné pozici

5 **6** **7** **8** **A** [1.0] ... 1 číslice správně

1 **2** **3** **9** **A** [1,2] ... 1 číslice správně, 2 číslice na nesprávných pozicích

9 **2** **3** **8** **A** [1.2] ... 1 číslice správně, 2 číslice na nesprávných pozicích

2 **9** **3** **5** **A** [0.4] ... 4 číslice na nesprávných pozicích

5 **2** **9** **3** **A** [4.0] ... úspěch, číslo je 5293

ML-45 (LE-12) Hra Acey-Deucey

?Num3 ?Bank
Start Num1 Num2 Odds Bet

Ve hře ML-45 Acey-Deucey sážíte na výskyt čísla v intervalu. Program vygeneruje dvě náhodná čísla Num1 a Num2. Podle čísel vám nabídne šanci na výhru (násobek sázky). Po vsazení vaší sázky program vypočítá třetí číslo Num3. Pokud Num3 leží v intervalu Num1 až Num2, vyhráváte sázku vynásobenou šancí. Pokud nevyhrajete, o sázku přijdete.

Lbl A Start nové hry s bankem 1000

Lbl B Vygenerování prvního čísla Num1 = 1...1000

Lbl C Vygenerování druhého čísla Num2 = 1...1000 (Num2 >= Num1)

Lbl D Výpočet šance (násobek sázky)

Lbl E Vložení sázky

Lbl B' Zobrazení minulého třetího čísla Num3 = 1...1000

Lbl C' Zobrazení stavu banku

*Note: Pro reprodukovatelnost náhody je možné inicializovat generátor náhody operací **Op 52**. Běžně není potřeba generátor náhody inicializovat.*

Example:

Pgm 45 ... aktivace knihovního programu ML-45

Op 52 ... inicializace generátoru náhody - pouze pro demonstraci

A [1000] ... start nové hry s bankem 1000

B [1] ... vygenerování prvního čísla Num1 = 1

C [118] ... vygenerování druhého čísla Num2 = 118

D [7.55] ... určení šance = 7.55

25E [-25] ... sázka = 25, prohra

B' [825] ... třetí číslo bylo Num3 = 825

C' [975] ... nový stav banku je 975

B [38] ... vygenerování prvního čísla Num1 = 38

C [636] ... vygenerování druhého čísla Num2 = 636

D [0.67] ... určení šance = 0.67

100E [67.22] ... sázka = 100, výhra $100 \cdot 0.6722 \dots = 67.22 \dots$

B' [181] ... třetí číslo bylo Num3 = 181

C' [1042.22] ... nová výše banku je 1042.22

ML-46 (LE-13) Hra Craps

?Last ?Bet

Roll Bet Start ?Bank Dice

Ve hře ML-46 Craps házíte dvěma kostkami. Hodíte-li při prvním hodu součet 7 nebo 11, vyhráváte. Hodíte-li 2, 3 nebo 12, prohráváte. V ostatních případech pokračujete dalším hodem.

Hodíte-li v dalším hodu 7, prohráváte. Hodíte-li znovu stejné číslo jako při prvním hodu, vyhráváte. Jinak pokračujete dalším hodem.

Hod kostkami je indikován číslem M.N, kde M a N je číslo na kostce 1...6. Důležitý je pouze součet čísel na kostkách, ne jejich pořadí.

Lbl A Hod kostkami

Lbl A' Zobrazení minulého hodu

Lbl B Zadání sázky

Při výhře se sázka přičte k banku, při prohře se odečte. Nebude-li sázka změněna, platí minulé výše sázky.

Lbl B' Zobrazení zvolené sázky

Lbl C Start nové hry s bankem 1000

Lbl D Zobrazení banku

Lbl E Hod jednou kostkou 1...6 (neovlivní stav hry)

*Note: Pro reprodukovatelnost náhody je možné inicializovat generátor náhody operací **Op 52**. Běžně není potřeba generátor náhody inicializovat.*

Example:

Pgm 46 ... aktivace knihovního programu ML-46

0 Op 52 ... inicializace generátoru náhody - pouze pro demonstraci

C ... Start nové hry s bankem 1000

2 5 B ... nastavení sázky = 25

A [1.1] ... první hod $1+1 = 2$, prohra

D [975] ... zobrazení nové výše banku = 975

A [5.1] ... první hod $5+1 = 6$

A [4.2] ... další hod $4+2 = 6$, výhra

D [1000] ... zobrazení nové výše banku = 1000

A [1.3] ... první hod $1+3 = 4$

A [4.3] ... další hod $4+3 = 7$, prohra

D [975] ... zobrazení nové výše banku = 1000

A [4.6] ... první hod $4+6 = 10$

A [4.4] ... další hod $4+4 = 8$

A [2.2] ... další hod $2+2 = 4$

A [1.6] ... další hod $1+6 = 7$, prohra

A [4.5] ... první hod $4+5 = 9$

A [5.3] ... další hod $5+3 = 8$

A [5.6] ... další hod $5+6 = 11$

5 0 0 B ... nastavení nové sázky = 500

A [5.4] ... další hod $5+4 = 9$, výhra

D [1450] ... zobrazení nové výše banku = 1450

ML-47 (LE-14) Hra Přistání na Marsu

?Burn

Burn Fuel Vel Alt Start

Hra ML-47 simuluje přistání na Marsu. Hru začínáte s výškou 2603 stop, rychlostí klesání 487 stop za sekundu a se zásobou paliva 630. Pro úspěšné přistání musí být rychlost maximálně 6, kterou ještě jsou schopny tlumiče utlumit. Ztratíte-li palivo nad zemí, modul přejde ve volný pád.

Zpočátku spálení jednotky paliva zajistí zrychlení 1 stopa/sec². S klesající hmotností modulu účinnost motorů roste. Gravitace na Marsu je 13 stopa/sec².

Lbl A Zážeh motorů se zadanou spotřebou paliva 0 až 75

Lbl A' Zobrazení posledního zážehu motorů

Lbl B Zobrazení zásoby paliva 'f'

Lbl C Zobrazení rychlosti 'v'

Lbl D Zobrazení výšky 'h'

Lbl E Start nové hry

Example:

Pgm 47 ... aktivace knihovního programu ML-47

E ... start nové hry, rychlost klesání -487, výška 2603, palivo 630

7 5 A ... zážeh 75, v = -422, h = 2149

5 0 A ... zážeh 50, v = -381, h = 1747

5 0 A ... zážeh 50, v = -339, h = 1387

5 0 A ... zážeh 50, v = -296, h = 1069

2 5 A ... zážeh 25, v = -280, h = 782

B [380] ... zobrazení zásoby paliva = 380

2 **5** **A** ... zážeh 25, $v = -263$, $h = 510$

7 **5** **A** ... zážeh 75, $v = -184$, $h = 286$

7 **5** **A** ... zážeh 75, $v = -101$, $h = 144$

5 **0** **A** ... zážeh 50, $v = -47$, $h = 70$

B [155] ... zobrazení zásoby paliva = 155

2 **5** **A** ... zážeh 25, $v = -26$, $h = 34$

2 **0** **A** ... zážeh 20, $v = -11$, $h = 15$

1 **3** **A** ... zážeh 13, $v = -6$, $h = 6$

1 **5** **A** ... zážeh 15, $v = +2$, $h = 4$

B [82] ... zobrazení zásoby paliva = 82

6 **A** ... zážeh 6, $v = -3$, $h = 4$

1 **0** **A** ... zážeh 10, $v = -1$, $h = 2$

8 **A** ... hladké přistání s rychlostí dopadu $v = -2.82$

B [58] ... zobrazení zbylého paliva = 58

ML-48 Hra Nim

Turn N->Start

Při hře Nim začínáte s N kameny. Každý hráč odebere 1 až 3 kameny. Prohrává ten hráč, který odebere poslední kámen. Další hru začíná ten, kdo v poslední hře prohrál.

Note: Kalkulátor záměrně dělá občasné chyby v taktice, aby poskytl hráči větší šanci na výhru.

Lbl A Tah hráče 1 až 3 kameny

Lbl E Zadání počtu kamenů N a start nové hry

Example:

RST ... jen pro demonstraci, hru začne hráč

Pgm 48 ... aktivace knihovního programu ML-48

1 5 E ... start nové hry s 15 kameny

3 A ... hráč odebere 3 kameny

[3] ... kalkulátor odebere 3 kameny

[9] ... zbude 9 kamenů

2 A ... hráč odebere 2 kameny

[2] ... kalkulátor odebere 2 kameny

[5] ... zbude 5 kamenů

1 A ... hráč odebere 1 kámen

[3] ... kalkulátor odebere 3 kameny

[1] ... zbude 1 kámen

1 A ... hráč odebere 1 kámen, tím prohrál

ML-49 Měření reakčního času, stopky

React Start Pause

Program ML-49 slouží k měření reakční doby a k měření časových intervalů.

Po stisku **A** program chvíli čeká (po náhodný čas, aby okamžik nebyl předvídatelný), pak spustí měření času a úkolem je stisknout kterékoliv tlačítko kalkulátoru (kromě tlačítek **2nd**, **GTO** a **R/S**). Reakční dobu zobrazí v sekundách, s rozlišením 10 ms.

Po stisku **D** se spustí měření časového intervalu (stopky). Uplynulý čas se zobrazuje s rozlišením 0,01 sekundy, s maximální délkou intervalu 10 minut. Přesnost měření času není vysoká, odchylka může být i 20%.

Stiskem **E** se měření času zastaví na aktuální hodnotě (lap time). Měření času přitom v kalkulátoru probíhá dál. Opětovným stiskem **E** (nebo stiskem **R/S**) měření času pokračuje.

Lbl A Start nového měření reakčního času

Lbl D Start stopek. Max. měřitelný interval je 10 minut.

Lbl E Pozastavení stopek (lap time), pokračování v měření (též R/S).

ML-50 (LE-20) Hra Námořní bitva

?Range	?Bear	?Display	Clear
Range	Bear	Fire	Start

Z důvodu poškození z boje nemá vaše fregata funkční radar ani motor. Poslední informací bylo, že se ve vzdálenosti 15000 yardů blíží ponorka z neznámého směru, s počáteční rychlostí 30 uzlů. Rychlost 30 uzlů odpovídá zhruba urazené vzdálenosti 1000 yardů za minutu. Vaše fregata je vybavena 15 torpédy, schopnými palby 1 torpédo za minutu. Po výstřelu je schopna tajná rozvědka hlásit výsledek zásahu podle vzdálenosti od cíle:

- více než 500: Žádné poškození ponorky. Ponorka pluje přímým směrem k vám.

- 50 až 500: Částečné poškození ponorky. Rychlost ponorky se sníží o 6 uzlů (tj. 200 yardů za minutu, minimální rychlost je 6 uzlů) a ponorka se snaží uniknout změnou směru o 45°. Novým minutím o více než 500 ponorka pokračuje přímým směrem na vás.

- Méně než 50: Potopení ponorky.

Vaše fregata se potopí vystřelením všech torpéd, aniž potopíte ponorku, nebo pokud se ponorka dostane k vám blíže než na 500 yardů.

Při příští hře se mohou počáteční podmínky mírně lišit. Chcete-li začínat hru vždy se stejnými podmínkami, použijte před hrou funkci **E**.

Lb| A Nastavení vzdálenosti dopadu torpéda v yardech

Nastavením 0 se torpédo nevypálí, jen se přesune ponorka.

Lb| A' Zobrazení aktuálně nastavené vzdálenosti dopadu torpéda

Lb| B Nastavení směru torpéda ve stupních

Lb| B' Zobrazení aktuálně nastaveného směru torpéda

Lb| C Vypálení torpéda s nastaveným směrem a vzdáleností.

Každý krok zabere 1 herní minutu, za kterou se ponorka dostane blíže.

Stav se zobrazí jako číslo XXXX.NN, kde XXXX je vzdálenost od cíle a NN je počet zbylých torpéd. Blikání XXXX.00 znamená, že nezbyla žádná torpéda. 0.NN indikuje minutí cíle o více než 5000 yardů. Blikání XX.NN znamená potopení ponorky.

Lbl C' Zobrazení výsledku posledního útoku

Lbl E Start nové hry

Počáteční stav je 15000.15, tj. vzdálenost ponorky 15000 yardů a 15 zbylých torpéd.

Lbl E' Nulování registrů hry do výchozího stavu

Example:

Pgm 50 ... aktivace knihovního programu ML-50

E' ... nulování registrů hry, pouze pro účely demonstrace

E [15000.15] ... start nové hry, vzdálenost 15000, 15 torpéd

... očekávaná vzdálenost ponorky je $15000 - 1000 = 14000$ yardů

1 4 0 0 0 A ... nastavení vzdálenosti torpéda na 14000 yardů

9 0 B ... nastavení směru torpéda na 90 stupňů

C [0.14] ... minutí cíle o více než 5000 yardů

... očekávaná vzdálenost ponorky je $14000 - 1000 = 13000$ yardů

1 3 0 0 0 A ... nastavení vzdálenosti torpéda na 13000 yardů

1 8 0 B ... nastavení směru torpéda na 180 stupňů

C [0.13] ... minutí cíle o více než 5000 yardů

... očekávaná vzdálenost ponorky je $13000 - 1000 = 12000$ yardů

1 2 0 0 0 A ... nastavení vzdálenosti torpéda na 12000 yardů

2 7 0 B ... nastavení směru torpéda na 270 stupňů

C [478.12] ... minutí cíle o 478 yardů. Ponorka zpomalila na 24 uzlů,

tj. 800 yardů za minutu, a změnila směr o 45°.

... očekávaná vzdálenost ponorky je $12000 - 800/2 = 11600$ yardů

11600A ... nastavení vzdálenosti torpéda na 11600 yardů

272B ... nastavení směru torpéda na 272 stupňů

C [1266.11] ... minutí pod 5000 yardů, ponorka pokračuje v přímém směru s rychlostí 800 yardů za minutu

... očekávaná vzdálenost ponorky je $11600 - 800 = 10800$ yardů

10800A ... nastavení vzdálenosti torpéda na 10800 yardů

268B ... nastavení směru torpéda na 268 stupňů

C [425.10] ... minutí cíle o 425 yardů. Ponorka zpomalila na 18 uzlů, tj. 600 yardů za minutu, a změnila směr o 45° .

... očekávaná vzdálenost ponorky je $10800 - 600/2 = 10500$ yardů.

10500A ... nastavení vzdálenosti torpéda na 10500 yardů

265B ... nastavení směru torpéda na 265 stupňů

C [164.09] ... minutí cíle o 164 yardů. Ponorka zpomalila na 12 uzlů, tj. 400 yardů za minutu, a změnila směr o 45° .

... očekávaná vzdálenost ponorky je $10500 - 400/2 = 10300$ yardů

10300A ... nastavení vzdálenosti torpéda na 10300 yardů

264B ... nastavení směru torpéda na 264 stupňů

C [181.08] ... minutí cíle o 181 yardů. Ponorka zpomalila na 6 uzlů, tj. 200 yardů za minutu, a změnila směr o 45° .

... očekávaná vzdálenost ponorky je $10300 - 200/2 = 10200$ yardů

10200A ... nastavení vzdálenosti torpéda na 10200 yardů

261B ... nastavení směru torpéda na 261 stupňů

C [255.07] ... minutí cíle o 255 yardů. Ponorka pokračuje rychlostí 6 uzlů, tj. 200 yardů za minutu, a je odchýlená o 45° .

... očekávaná vzdálenost ponorky je $10200 - 200/2 = 10100$ yardů

10100A ... nastavení vzdálenosti torpéda na 10100 yardů

262B ... nastavení směru torpéda na 262 stupňů

C [176.06] ... minutí cíle o 176 yardů. Ponorka pokračuje rychlostí 6 uzlů, tj. 200 yardů za minutu, a je odchýlená o 45°.

... očekávaná vzdálenost ponorky je $10100 - 200/2 = 10000$ yardů

10000A ... nastavení vzdálenosti torpéda na 10000 yardů

263B ... nastavení směru torpéda na 263 stupňů

C [100.05] ... minutí cíle o 100 yardů. Ponorka pokračuje rychlostí 6 uzlů, tj. 200 yardů za minutu, a je odchýlená o 45°.

... očekávaná vzdálenost ponorky je $10000 - 200/2 = 9900$ yardů

9900A ... nastavení vzdálenosti torpéda je 9900 yardů

264B ... nastavení směru torpéda na 264 stupňů

C [bliká 26.05] ... zásah se vzdáleností 26 yardů, zbylo 5 torpéd