

# PROGRAMOVATELNÝ KALKULÁTOR

## ET-58



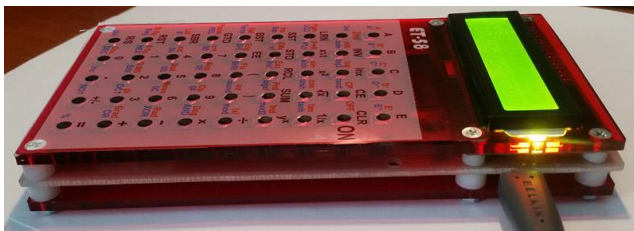
*uživatelská příručka*

# Programovatelný kalkulátor ET-58, uživatelská příručka

Verze příručky 1.0

říjen 2020

Vztaženo k verzi kalkulátoru 201005



webové stránky:

<http://www.eternal59.com/>

zrcadlo stránek:

<http://www.breatharian.eu/et58/>

# OBSAH

1. Mapa rozložení klávesnice	10
2. Charakteristika	12
3. Popis	14
4. Jak používat kalkulátor	15
5. Odchylky od TI-58/59	17
6. Formát čísel	22
7. Klávesnice	23
8. Indikátory na displeji	24
9. Indikace chyby	26
10. Editor čísla	27
11. Číselné výrazy	28
12. Nepřímé adresování	29
13. Programování	31
14. Tlačítka a instrukce	34
00 ... 09 Základní číslice, 0...9	34
0A ... 0F Hexadecimální číslice, 0A...0F	34
10 ... 1F Písmenné návěští, A...F	35
20 Vypnutí kalkulátoru, OFF	36
21 Alternativní funkce, 2nd	37
22 Inverze funkce, INV	37
23 Přirozený logaritmus a exponent, $\ln x$	38
24 Oprava chyby, CE	39
25 Vymazání displeje, CLR	39
26 Podprogram s nepřímou adresou, SBR Ind	40
27 Nepřímá interní instrukce, HIR Ind	40
28 Dekadický logaritmus a exponent, $\log$	41
29 Vymazání programu a registru T, CP	42
2B Zadání kódu instrukce, code	42
2C Dvojkový logaritmus a exponent, $\log_2$	43
2D Generátor náhodného čísla, rand	43
30 Tangens, tan	44
31 Programování, LRN	45
32 Záměna registrů X a T, $x \leftrightarrow t$	45

33 Druhá mocnina čísla, $x^2$	46
34 Druhá odmocnina čísla, $Vx$	46
35 Převrácená hodnota čísla, $1/x$	46
36 Výběr programu knihovny, Pgm	47
37 Převod kartézských a polárních souřadnic, $P \rightarrow R$	48
38 Sinus, sin	49
39 Kosinus, cos	49
3A Teplota, Temp	50
3B Záměna registrů X a Y, $x \leftrightarrow y$	50
3C Hyperbolický sinus, sinh	51
3D Hyperbolický kosinus, cosh	51
3E Hyperbolický tangens, tanh	52
40 Nepřímé adresování, Ind	52
41 Krok programu vpřed, SST	53
42 Uložení čísla do datového registru, STO	53
43 Vyvolání čísla z datového registru, RCL	53
44 Přičtení a odečtení čísla od datového registru, SUM	54
45 Mocnina a odmocnina, $y^x$	54
46 Vložení prázdného bajtu do programu, Ins	55
47 Vymazání datových registrů, CMs	55
48 Záměna čísla s datovým registrem, Exc	56
49 Vynásobení a vydělení datového registru, Prd	56
4A Zjištění napětí baterie, Bat	56
4B Faktoriál, $x!$	57
4C Přirozený logaritmus faktoriálu, $\ln x!$	58
4D Dekadický logaritmus faktoriálu, $\log x!$	58
4E Modulo floor, mod2	59
50 Absolutní hodnota, $ x $	60
51 Krok programu zpět, BST	60
52 Režim exponentu, EE	61
53 Levá závorka, (	61
54 Pravá závorka, )	61
55 Dělení, :	62
56 Zrušení bajtu z programu, Del	62
57 Technický mód, Eng	62
58 Zaokrouhlení, Fix	63
59 Celé číslo, Int	64
5A Nastavení kontrastu displeje, LCD	64
5B Posun doleva, SHL	65
5C Posun doprava, SHR	65
5D Zaokrouhlení, round	66
5E Modulo trunc, mod	67
60 Stupně, Deg	67
61 Skok, GTO	68

62	Nepřímý výběr programu knihovny, Pgm Ind .....	68
63	Nepřímá změna čísla s datovým registrem, Exc Ind .....	69
64	Nepřímé násobení a dělení registru, Prd Ind .....	69
65	Násobení, x .....	70
66	Prodleva, Pause .....	70
67	Rovno, x=t .....	70
68	Žádná operace, Nop .....	71
69	Speciální operace, Op .....	72
6A	Relativní skok, REL .....	72
6B	Nepřímá inkrementace/dekrementace registru, Inc Ind .....	73
6C	Nepřímá operace s registry, Reg Ind .....	73
6D	Nepřímá podmínka, IF Ind .....	74
6E	Bitový součin, AND .....	74
70	Radiány, Rad .....	75
71	Podprogram, SBR .....	76
72	Nepřímé uložení čísla do registru, STO Ind .....	76
73	Nepřímé vyvolání obsahu registru, RCL Ind .....	77
74	Nepřímé přičtení a odečtení čísla z registru, SUM Ind .....	77
75	Odečtení, - .....	78
76	Návěští, Lbl .....	78
77	Větší nebo rovno, x>=t .....	79
78	Statistika, Stat .....	79
79	Průměr, Mean .....	80
7A	Podmíněný skok, IF .....	81
7E	Bitový exkluzivní součet, XOR .....	84
80	Grady, Grad .....	84
81	Reset, RST .....	85
82	Interní instrukce, HIR .....	85
83	Nepřímý skok, GTO Ind .....	88
84	Nepřímá speciální operace, Op Ind .....	89
85	Přičtení, + .....	89
86	Nastavení a nulování přepínače, StFlg .....	90
87	Test přepínače, IfFlg .....	90
88	Převody minut a sekund, DMS .....	92
89	Ludolfovo číslo, pi .....	93
8A	Operace s registry, Reg .....	93
8B	Hexadecimální mód, HEX .....	95
8C	Binární mód, BIN .....	95
8D	Oktalový mód, OCT .....	96
8E	Bitový součet, OR .....	96
91	Start a stop programu, R/S .....	97
92	Návrat z podprogramu, RTN .....	97
93	Desetinná tečka, .....	98
94	Změna znaménka, +/- .....	98

95 Provedení výpočtu, =.....	98
97 Programová smyčka, Dsz .....	99
9A Zlatý řez, phi.....	101
9B Dekadický mód, DEC .....	101
9C Inkrementace a dekrementace registru, Inc .....	102
9D Bitová inverze, NOT .....	103
9E Procenta, %.....	103
<b>15. Speciální operace Op</b> .....	<b>105</b>
Op 00 Vymazání tiskových registrů 1 až 4 .....	105
Op 01..04 Nastavení tiskového registru 1..4 .....	105
Op 09 Načtení knihovního programu .....	106
Op 0A Výstup registrů 1 a 2 na 1. řádek za běhu .....	106
Op 0B Výstup registru 1 s X na 1. řádek za běhu .....	106
Op 0C Výstup půl-registru 1 s X na 1. řádek za běhu .....	107
Op 0D Výstup registrů 3 a 4 na 2. řádek za běhu .....	107
Op 0E Výstup registru 3 s X na 2. řádek za běhu .....	108
Op 0F Výstup půl-registru 3 s X na 2. řádek za běhu .....	108
Op 10 Funkce sign .....	109
Op 11 Variace.....	109
Op 12 Koeficienty lineární regrese .....	110
Op 13 Korelační koeficient .....	111
Op 14 Lineární regrese Y z X.....	111
Op 15 Lineární regrese X z Y .....	112
Op 16, Op 17 Organizace paměti.....	112
Op 18 Nastavení přepínače 7 bez chyby .....	112
Op 19 Nastavení přepínače 7 při chybě.....	112
Op 1A Výstup registrů 1 a 2 na 1. řádek při zastavení .....	113
Op 1B Výstup registru 1 s X na 1. řádek při zastavení .....	113
Op 1C Výstup půl-registru 1 s X na 1. řádek při zastavení .....	113
Op 1D Aktivace módu displeje 'Přepínače' .....	114
Op 1E Aktivace módu displeje 'Registr T' .....	114
Op 1F Nastavení módu displeje 'Text' .....	114
Op 20 až Op 2F Inkrementace datového registru .....	114
Op 30 až Op 3F Dekrementace datového registru .....	115
Op 40 Vstup klávesy z klávesnice .....	115
Op 41 Test stisku tlačítka .....	115
Op 42 Zobrazení jednoho znaku na displeji .....	116
Op 43 Načtení fontu .....	117
Op 44 Prodleva.....	120
Op 45 Zobrazení ukazatele zleva na 1. řádku za běhu.....	121
Op 46 Zobrazení ukazatele zleva na 2. řádku za běhu.....	121
Op 47 Zobrazení textu s ukazatelem zleva při zastavení .....	122
Op 48 Zobrazení ukazatele zprava na 1. řádku za běhu .....	123

Op 49 Zobrazení ukazatele zprava na 2. řádku za běhu .....	123
Op 4A Zobrazení textu s ukazatelem zprava při zastavení.....	124
Op 4B Zobrazení sloupce grafu za běhu.....	125
Op 4C Zobrazení sloupce grafu s textem po zastavení .....	126
Op 4D Nastavení pixelu.....	127
Op 4E Vymazání pixelu .....	127
Op 4F Přepnutí pixelu.....	127
Op 50 Vyhledání největšího společného dělitele .....	128
Op 51 Načtení registru generátoru náhody .....	128
Op 52 Nastavení registru generátoru náhody .....	129
Op 53 Načtení předdefinovaného textu.....	129
Op 54 Přidání čísla k textu .....	130
Op 55 Inicializace zásobníku komplexních čísel a zlomků .....	131
Op 56 Zjištění počtu čísel v zásobníku komplexních čísel.....	132
Op 57 Vložení čísla do zásobníku komplexních čísel .....	132
Op 58 Načtení čísla ze zásobníku komplexních čísel.....	132
Op 59 Zrušení čísla ze zásobníku komplexních čísel .....	132
Op 5A Záměna dvou čísel v zásobníku komplexních čísel.....	133
Op 5B Duplikace čísla v zásobníku komplexních čísel .....	133
Op 5C Součet dvou komplexních čísel $X+Y$ .....	133
Op 5D Rozdíl dvou komplexních čísel $X-Y$ .....	133
Op 5E Násobek dvou komplexních čísel $X*Y$ .....	133
Op 5F Podíl dvou komplexních čísel $X/Y$ .....	133
Op 60 Umocnění dvou komplexních čísel $X^AY$ .....	134
Op 61 Odmocnění dvou komplexních čísel $X^{(1/Y)}$ .....	134
Op 62 Logaritmus dvou komplexních čísel $\log Y(X)$ .....	134
Op 63 Druhá mocnina komplexního čísla $X^2$ .....	134
Op 64 Druhá odmocnina komplexního čísla $VX$ .....	134
Op 65 Převrácená hodnota komplexního čísla $1/X$ .....	135
Op 66 Přirozený exponent komplexního čísla $e^AX$ .....	135
Op 67 Přirozený logaritmus komplexního čísla $\ln(X)$ .....	135
Op 68 Sinus komplexního čísla $\sin(X)$ .....	135
Op 69 Kosinus komplexního čísla $\cos(X)$ .....	135
Op 6A Tangens komplexního čísla $\tan(X)$ .....	135
Op 6B Arkus sinus komplexního čísla $\text{asin}(X)$ .....	135
Op 6C Arkus kosinus komplexního čísla $\text{acos}(X)$ .....	136
Op 6D Arkus tangens komplexního čísla $\text{atan}(X)$ .....	136
Op 6E Konverze komplexního čísla na polární .....	136
Op 6F Konverze polárního čísla na komplexní .....	136
Op 70 Vyhledání průchodu nulou funkce $A'$ .....	136
Op 71 Simpsonův integrál funkce $A'$ .....	137
Op 72 Konverze úhlu z aktuální úhlové jednotky na radiány .....	137
Op 73 Konverze úhlu z radiánů na aktuální úhlovou jednotku .....	138
Op 74 Normální distribuce pravděpodobnosti $Z(x)$ .....	138

Op 75 Komplementární Gaussova distribuce $Q(x)$ CGD .....	138
Op 76 Kumulativní normální distribuce $P(x)$ CND .....	138
Op 77 Maximum .....	138
Op 78 Minimum .....	138
Op 79 Vynulování všech HIR registrů .....	138
Op 7A Konverze desetinného čísla na zlomek .....	139
Op 7B Převod zlomku na desetinné číslo .....	139
Op 7C Součet zlomků $X+Y$ .....	139
Op 7D Rozdíl zlomků $X-Y$ .....	139
Op 7E Násobek dvou zlomků $X*Y$ .....	139
Op 7F Podíl dvou zlomků $X/Y$ .....	140
Op 80 Krátká prodleva 10 msec .....	140
Op 81 Krátká prodleva 100 msec .....	140
Op 82 Odstranění skrytých číslic .....	140
Op 83 Zahájení měření času .....	140
Op 84 Zjištění uplynulého času .....	141
Op 85 Zjištění počtu datových registrů .....	141
Op 86 Zjištění stavu uživatelských přepínačů .....	141
Op 87 Výpočet kontrolního součtu ROM paměti .....	141
Op 88 Nastavení prodlevy pro vypnutí kalkulátoru .....	142
Op 89 Zjištění prodlevy pro vypnutí kalkulátoru .....	142
Op 8A Zobrazení verze firmware kalkulátoru .....	142
Op 8B Reset kalkulátoru .....	142
16. Tabulka znaků .....	143
17. Knihovní programy .....	144
ML-01 Diagnostika .....	145
ML-02 Determinant matice .....	147
ML-03 Sčítání a násobení matic .....	151
ML-04 Komplexní aritmetika .....	155
ML-05 Komplexní funkce .....	158
ML-06 Komplexní trigonometrické funkce .....	161
ML-07 Vyčíslení polynomu .....	164
ML-08 Průchod funkce nulou .....	165
ML-09 Simpsonův integrál funkce .....	168
ML-10 Simpsonův diskrétní integrál .....	170
ML-11 Řešení trojúhelníků zadaných stranami .....	171
ML-12 Řešení trojúhelníků zadaných úhly .....	174
ML-13 Kruhový oblouk .....	177
ML-14 Normální rozdělení .....	179
ML-15 Generátor náhodných čísel .....	181
ML-16 Variace, permutace, kombinace, faktoriál .....	182
ML-17 Klouzavý průměr .....	187
ML-18 Složený úrok .....	188



ML-19 Splátky.....	191
ML-20 Den v týdnu, dny mezi daty .....	195
ML-21 Hra HI-LO .....	197
ML-22 Běžný a spořicí účet.....	198
ML-23 DMS operace .....	201
ML-24 Konverze jednotek 1 .....	203
ML-25 Konverze jednotek 2 .....	204
ML-26 Aritmetika zlomků .....	205
ML-27 Astabilní generátor s obvodem 555 .....	209
ML-28 (EE-07) Konverze poměrů .....	212
ML-29 (EE-11) Reaktance LC .....	213
ML-30 (EE-12) Konverze sériové/paralelní impedance .....	215
ML-31 (EE-13) Aktivní filtr .....	217
ML-32 (EE-14) Pasivní dolní propust .....	221
ML-33 (EE-15) Konvoluce signálu.....	224
ML-34 (EE-17) Diskrétní Fourierova transformace .....	227
ML-35 Ohmův zákon .....	230
ML-36 Sérové a paralelní řazení .....	232
ML-37 (AV-23) Konverze časové zóny.....	233
ML-38 (MU-06) Shellovo třídění .....	236
ML-39 (MU-09) Rozklad na prvočinitele.....	238
ML-40 (MU-21) Aritmetika s proměnnými .....	239
ML-41 (MU-14) Interpolace .....	240
ML-42 (MU-16) Vyhledání minima a maxima.....	242
ML-43 Kontrolní součet .....	244
ML-44 (LE-09) Hra Codebreaker.....	248
ML-45 (LE-12) Hra Acey-Deucy .....	249
ML-46 (LE-13) Hra Craps.....	251
ML-47 (LE-14) Hra Přistání na Marsu .....	253
ML-48 Hra Nim .....	255
ML-49 Měření reakčního času, stopky .....	256
ML-50 (LE-20) Hra Námořní bitva .....	257

# 1. Mapa rozložení klávesnice

U každého tlačítka je uveden na 1. řádku základní význam, na 2. řádku první alternativní význam (po stisku tlačítka **2nd**) a na 3. řádku druhý alternativní význam (po dvojnásobném stisku tlačítka **2nd** **2nd**, tj. **3rd**).

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

	A	B	C	D	E	
A'	●	B'	●	D'	●	E'
A''	●	B''	●	D''	●	E''
2nd	INV	Inx	CE	CLR	ON	
3rd	HIR code	log log2	CP rand	OFF		
LRN	x↔t	x <sup>2</sup>	√x	1/x		
Pgm Temp	P→R x↔y	sin sinh	cos cosh	tan tanh		
SST	STO	RCL	SUM	y <sup>x</sup>		
Ins Bat	CMs x!	Exc Inx!	Prd logx!	Ind mod2		
BST	EE	(	)	÷		
Del LCD	Eng SHL	Fix SHR	Int round	x  mod		
GTO	7	8	9	x		
Pause REL	x=t 0D	Nop 0E	Op 0F	Deg AND		
SBR	4	5	6	-		
Lbl IF	x≥t 0A	Stat 0B	Mean 0C	Rad XOR		
RST	1	2	3	+		
StFlg Reg	IfFlg HEX	D.MS BIN	pi OCT	Grad OR		
R/S	0	.	+/-	=		
Write phi	Dsz DEC	Adv Inc	Prt NOT	Lst %		

## 2. Charakteristika

- Procesor ATmega328P (4 MHz, 32 KB ROM, 2 KB RAM)
- Napájecí napětí 2.5 V (z baterie) až 5.5 V (z USB konektoru).
- Přesnost výpočtů 19 číslic
- Zobrazení údaje až na 14 platných číslic
- Exponent 4 číslice, rozsah +- 9863
- 1000 programových kroků uživatelského programu v EEPROM (uchování programu i bez baterie)
- 110 datových registrů
- 16 řídicích HIR registrů
- Dvouřádkový LCD displej (2 x 16 alfanumerických znaků)
- 45 tlačítek
- Kód kalkulátoru kompletně napsaný v AVR assembleru
- Vestavěná knihovna s 50 programy a s délkou téměř 10000 kroků
- Exponenciální a logaritmické funkce
- Trigonometrické funkce
- Hyperbolické funkce
- Faktoriál desetinných a velkých čísel
- Generátor náhodných čísel
- Indexový přístup k proměnným
- Nepřímé parametry funkcí
- Vědecký a technický mód zobrazení s exponentem
- Zobrazení HEX, OCT a BIN, včetně desetin
- Bitové operace AND, OR, XOR, NOT, posuny
- Pseudografické zobrazení grafů a ukazatelů
- Programové zobrazení textu
- Dynamický vstup z klávesnice za běhu programu
- Statistické funkce a lineární regrese
- Absolutní adresování, návěští, relativní skoky
- Výpočty s maticemi
- Komplexní čísla
- Zlomky
- Vyčíslení polynomů
- Numerické hledání kořenů funkce
- Numerický výpočet integrálů
- Interpolace a aproximace
- Výpočty trojúhelníků
- Převody jednotek

- Kruhová výseč
- Kombinace, permutace
- Plovoucí průměr
- Úroky a splátky
- Časové převody, časové zóny
- Interval mezi daty, den v týdnu
- Hry (Hi-Lo, Codebreaker, Acey-Deucey, Lander a další)
- Astabilní generátor s 555
- Reaktance kapacitorů a induktorů
- Sériové a paralelní řazení součástek
- Výpočty aktivních a pasivních filtrů
- Konvoluce signálu
- Diskrétní Fourierova transformace
- Ohmův zákon
- Třídění čísel a medián
- Hledání prvočinitelů
- Hledání minima a maxima funkce
- Měření reakční doby

### 3. Popis

Kalkulátor ET-58 je určen především kutilům a zájemcům o retro výpočetní techniku. Konceptně vychází z populárního kalkulátoru TI-58/59, vyvinutého v roce 1978 firmou Texas Instruments. Jeho vlastnosti rozšiřuje, a přitom se snaží maximálně zachovat funkčnost původního kalkulátoru. Kalkulátor ET-58 je nabízen ve formě stavebnice. Zaměřuje se na co nejlevnější provedení a snadné sestavení stavebnice, proto používá maximálně jednoduchou konstrukci bez hmatníků tlačítek a s minimem SMD součástek.

## 4. Jak používat kalkulátor

Kalkulátor ET-58 je opatřen 2-řádkovým alfanumerickým LCD displejem, 45 mikrospláči, procesorem a napájecí baterií.

Kalkulátor se zapne stiskem tlačítka **CLR** (alternativní funkce **OFF**). Vypne se buď tiskem **OFF** (tj. stiskem tlačítek **2nd CLR**), nebo automaticky po zvolené době neaktivity kalkulátoru (implicitně nastaveno na 30 sekund - lze změnit pomocí **Op 88**). Nebude-li po dobu delší než několik minut vyjmuta baterie (a odpojeno externí napájení), zůstane v kalkulátoru zachován i obsah paměťových registrů, zobrazené číslo na displeji a započaté operace (tj. obsah RAM paměti). U kalkulátoru bez vložené baterie nestiskejte tlačítko **CLR** chcete-li zachovat obsah RAM paměti po zapnutí, jinak se kalkulátor po zapnutí resetuje a obsah RAM zapomene. Ztráta obsahu paměti se netýká programu v ROM (knihovny) a EEPROM (uživatelský program), ty zůstávají v nezměněném stavu i bez baterie.

Dojde-li k vyjmutí baterie u nevypnutého kalkulátoru, může dojít k okamžité ztrátě obsahu RAM paměti. Tímto způsobem je možné kalkulátor resetovat a uvést do výchozího stavu, vykazuje-li nesprávné chování nebo chcete-li nastavit kalkulátor do výchozího stavu. Navrácení kalkulátoru do výchozího nastavení je možné provést též podprogramem CE knihovního programu 1 (vyvolání: **Pgm 0 1 SBR CE RST**).

Obsah uživatelského programu ani nahraná uživatelská knihovna nebudou ztraceny ani po dlouhém vyjmutí baterie.

Pozor - nevyjímejte baterii a neodpojujte napájení během probíhajících operací zápisu do programu nebo během nahrávání knihovny do paměti. Mohlo by dojít k poškození obsahu paměti.

Kontrast LCD displeje je závislý na napájecím napětí. K nastavení kontrastu displeje slouží tlačítko **LCD** (vyvoláte ho stiskem tlačítek **2nd 2nd BST**). Po stisku tlačítka zadejte do kalkulátoru číslo **0** až **9**, které představuje požadovaný kontrast LCD displeje. Číslo 0 znamená minimální kontrast (znaky na displeji jsou šedé nebo dokonce sotva viditelné), číslo 9 znamená maximální kontrast (pod znaky na displeji je tmavý obdélník). Změnou kontrastu displeje se můžete dostat až mimo zobrazitelný rozsah, kdy už nerozeznáte text na displeji. I v takovém případě zkuste změnit kontrast tlačítkem **LCD**. Nejsou-li na displeji viditelné žádné znaky

(kalkulátor vypadá jako vypnutý), zkuste z klávesnice nastavit vyšší kontrast displeje použitím vyššího čísla. Případně stiskněte nejdříve **CLR**, možná je kalkulátor ve vypnutém stavu. Naopak jsou-li na displeji černé obdélníky, nastavte nižší hodnotu kontrastu displeje. Nevíte-li, v jakém stavu kalkulátor zůstal naposledy (možná je nyní v programovacím módu), vyjměte baterku, stiskněte **CLR**, čímž kalkulátor resetujete. Pak vložte baterii, zkuste kalkulátor zapnout tlačítkem **CLR** a provést znovu nastavení kontrastu displeje. V případě neúspěchu zkuste novou baterii nebo externí napájení, možná se jen vybila baterie.

Po resetu kalkulátoru (při prvním zapnutí po vložení baterie) se na displeji kalkulátoru 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. Chcete-li zobrazit verzi firmware znovu, vyjměte a opět zasuněte baterii do kalkulátoru, aniž kalkulátor vypnete. Tímto postupem dosáhnete reset kalkulátoru. Vyjmete-li baterii z kalkulátoru ve vypnutém stavu, kalkulátor si může uchovat data i hodinu, aniž provede po zapnutí reset - v tom případě stiskněte tlačítko **CLR** a k resetu dojde i při vypnutém stavu. K zobrazení verze firmware můžete též použít operaci **Op 8A** nebo program **Pgm 01 A**.

Při každém zapnutí kalkulátor provádí kontrolu integrity vnitřní paměti ROM pomocí 16-bitového kontrolního součtu (CRC-XModem). Objeví-li se interní chyba paměti, kalkulátor na 2 sekundy zobrazí varování "CRC Error". Kalkulátor je sice možné nadále používat, ale jeho procesor je zjevně vadný a může vykazovat nepředvídatelnou činnost.



## 5. Odchylyky od TI-58/59

Přestože se software kalkulátoru ET-58 snaží o maximální kompatibilitu s původní řadou TI-58/59, nelze zajistit plnou slučitelnost a může být nutné některé programy při importu upravovat.

### Vyšší přesnost

Původní kalkulátory TI-58/59 pracovaly s čísly v BCD kódu (1 bajt = 2 dekadické číslice) a s interní přesností 13 číslic. Velikost čísla byla 10 bajtů. Kalkulátor ET-58 používá sice také číslo o velikosti 10 bajtů, ale používá binární kód, což mu umožňuje dosáhnout vyšší přesnosti mantisy 19 číslic a rychlejší výpočty. Vyšší přesnost obvykle není na závadu, ovšem použití binárního kódu namísto BCD kódu se může projevit ztrátou přesnosti u čísel s konečným počtem číslic. Kalkulátor se snaží korigovat odchylky, ale někdy se mohou projevit a může být nutné s nimi počítat.

Typickým příkladem jsou celá čísla použitá jako čítač ve smyčkách. Po tisících cyklech přičtení/odečtení hodnoty 1 může vzniknout malá odchylka od celého čísla, která se sice neobjeví na displeji, ale může se projevit při porovnání čísla na shodu. Kalkulátor tuto odchylku ošetřuje tak, že funkce smyčky (DSZ, DJNZ, DJZ) po každé dekrementaci číslo zaokrouhlí na celé číslo.


Přestože kalkulátor zohledňuje určitou toleranci odchylky, může být u opakovaných výpočtů potřeba počítat s rostoucí odchylkou od předpokládané přesné hodnoty a před porovnáním na shodu použít zaokrouhlení na celé číslo či porovnávat v intervalu s tolerancí.

Dále se odlišná přesnost projeví v generátoru náhodných čísel. Původní generátor náhodných čísel TI-58/59 provádí operace sčítání a násobení a posouvá oblast skrytých číslic na viditelné pozice. To se projeví tím, že výpočet náhodného čísla se u ET-58 po několika krocích zcela odchýlí od generátoru náhody původního kalkulátoru TI-58/59. Zpravidla by tato skutečnost neměla vadit. Navíc v programech ET-58 se původní generátor náhody již nepoužívá, kalkulátor má k dispozici vnitřní generátor náhody s podstatně kvalitnějším rozložením náhodnosti.

### Hexadecimální kód

Kód programu je u původního kalkulátoru TI-58/59 uchován v BCD kódu.

Každý bajt je rozdělen na poloviny obsahující 2 číslice s hodnotou 0 až 9. Dvě číslice tak představují dekadickou hodnotu 0 až 99. Naproti tomu kód kalkulátoru ET-58 je v paměti uložen v hexadecimálním kódu. Je vyjádřen opět pomocí 2 číslic, ale v rozsahu 0 až 15. Číslice s hodnotou 10 až 15 jsou vyjádřeny písmenem A až F. Hexadecimální zápis podporuje jak původní význam, tj. zápis hodnoty 00 až 99 (tj. 100 hodnot) pomocí 2 číslic 0 až 9, tak i hexadecimální význam s hodnotou 00 až FF (tj. 256 hodnot). Interpretace kódu závisí na významu bajtu.

Původní kódy tlačítek jsou zachovány. Např. tlačítko  má i zde kód 25, stejně jako u původní TI-58/59, bez ohledu na to, zda jde o binární nebo hexadecimální zápis. Hexadecimální znaky množinu kódů rozšiřují o kódy se znaky A až F.

Absolutní adresa a číslo registru se vyjadřuje pomocí 2 dekadických číslic 0 až 9, stejně jako u původního TI-58/59, a má hodnotu 0 až 99. U čísel registrů kalkulátoru lze jako první znak použít i číslici A a tím rozšířit počet adresovatelných registrů ze 100 na 110.



V některých speciálních případech má kód programu význam 2 hexadecimálních číslic. Jedná se např. o kód HIR instrukce. První číslice 0 až F představuje kód operace, druhá číslice 0 až F představuje číslo HIR registru 0 až 15.

## Zaokrouhlení smyček

Smyčka DSZ u původního kalkulátoru sníží hodnotu registru o 1 a nedosáhne-li nuly, provede zpětný skok. Jak už bylo zmíněno výše, díky binárnímu vyjádření čísla se může hodnota registru u ET-58 mírně odchýlit od celého čísla, a to by vedlo k nepřesnosti smyčky. Kalkulátor to ošetřuje tak, že po každé dekrementaci výsledek navíc ještě zaokrouhlí na celé číslo. To se může projevit v případě, že původní program z TI-58/59 počítá s používáním desetinných čísel.

To stejné platí u instrukcí DJNZ a DJZ (skupina HIR instrukcí), které však původní kalkulátor nemá.

## Opakování výpočtu

Kalkulátor ET-58 po stisku klávesy  opakuje naposledy zadanou aritmetickou operaci. Některé programy původní TI-58/59 s opakováním operací nepočítají, použijí klávesu  vícekrát a tím může vzniknout chybný

výpočet.

## Vyšší rychlost

Kalkulátor ET-58 používá modernější a výkonnější procesor, s vyšší taktovací frekvencí. Díky tomu výpočty probíhají znatelněji rychleji než u původního kalkulátoru. Tato vlastnost zpravidla není na závadu. Může se projevit u programů počítajících s rychlostí původního kalkulátoru, jako např. program pro test postřehu. Ale takových programů je zanedbatelné množství.

## HIR registry

U původního kalkulátoru TI-58/59 jsou k dispozici speciální interní instrukce HIR, používané vnitřním kódem a nezveřejněné v uživatelské příručce. Instrukce HIR původně používaly registry aritmetických operací 0 až 9. U kalkulátoru ET-58 jsou instrukce HIR legalizované jako platné instrukce a jsou dokonce využívány knihovnamí jako hlavní pracovní registry. Na rozdíl od původního kalkulátoru nejsou sdíleny s aritmetickou jednotkou. Jedná se o plnohodnotné registry, nepřepisované žádnými jinými operacemi.

## Zaokrouhlení Fix

U původního kalkulátoru TI-58/59 slouží klávesa **Fix** k nastavení počtu zobrazených desetinných míst na 0 až 8. Hodnota 9 vypíná zaokrouhlení, číslo se zobrazí na plný počet míst. Stejný význam má **INV Fix**.

Kalkulátor ET-58 umožňuje zobrazit číslo až na 13 desetinných míst. Proto je parametr instrukce Fix rozšířen na hodnoty 0 až 9 a A až D, představující zaokrouhlení na 0 až 13 desetinných míst. Kód **Fix 0F** slouží k vypnutí zaokrouhlení, stejně tak jako povel **INV Fix**.

U starších programů je kód **Fix 9** často používán k vypnutí zaokrouhlení. V takových případech je nutné kód změnit na **INV Fix** nebo **Fix 0F**.

## Indikace chyby

U původního kalkulátoru TI-58/59 je někdy používáno blikání displeje k indikaci chyby programu. Pro tento účel bývá využíván kód **0 1/X**.

Takový kód bude u ET-58 fungovat beze změny.

Druhým případem, někdy používaným, je kód  $\boxed{+}$   $\boxed{=}$ . U původního kalkulátoru tato posloupnost zajistí rozblikání údaje zobrazeného na displeji. U ET-58 neaktivuje tato posloupnost indikaci chyby. Indikaci lze nahradit buď posloupností  $\boxed{X/T}$   $\boxed{0}$   $\boxed{1/X}$   $\boxed{X/T}$  nebo použít  $\boxed{\text{StFlg}}$   $\boxed{0F}$ , který zapne přepínač 15, který je současně stavem indikace chyby.

## Přenos INV

U původního kalkulátoru TI-58/59 je někdy využíváno nastavení prefixu  $\boxed{\text{INV}}$  (inverzní funkce) přes několik instrukcí. Kalkulátor počítá s tím, že instrukce tento prefix nevyužívající jeho stav nezmění. U kalkulátoru ET-58 jsou mnohé funkce rozšířené o inverzní nebo alternativní funkci, a tak většina povelů kód  $\boxed{\text{INV}}$  zpracuje a neuchová.

Nejčastějším využitím přenosu  $\boxed{\text{INV}}$  je u návěstí. Za prvním návěstím je samotný povel  $\boxed{\text{INV}}$ , za kterým následuje druhé návěstí a první příkaz. V takovém případě se spuštěním programu z prvního návěstí vyvolá inverzní funkce příkazu, spuštěním programu z druhého návěstí se vyvolá neinvertovaná funkce. V tomto případě je stav tlačítka  $\boxed{\text{INV}}$  zachován během návěstí  $\boxed{\text{LBL}}$  i u kalkulátoru ET-58.

## Pořadí 2nd INV

U původního TI-58/59 nezáleželo na pořadí stisku tlačítek  $\boxed{2\text{nd}}$  a  $\boxed{\text{INV}}$ . V obou případech se provedla inverzní alternativní funkce. Při zápisu do programu se v jednom případě uložil kód INV a alternativní kód tlačítka, v druhém případě se uložil kód  $\boxed{2\text{nd}}$   $\boxed{\text{INV}}$  (kód 26) a normální kód tlačítka - změna na alternativní tlačítko se provedla až během provádění programu.

U kalkulátoru ET-58 toto nelze použít a je nutné zachovat pořadí stisku tlačítek  $\boxed{\text{INV}}$   $\boxed{2\text{nd}}$   $\boxed{\dots}$ . Tlačítko  $\boxed{\text{INV}}$  nejdříve nastaví prefix inverzní funkce a tlačítko  $\boxed{2\text{nd}}$   $\boxed{\dots}$  uloží do programu alternativní kód tlačítka. Při opačném stisku,  $\boxed{2\text{nd}}$   $\boxed{\text{INV}}$ , se provede alternativní funkce tlačítka  $\boxed{\text{INV}}$ , tj. funkce  $\boxed{\text{HIR}}$ .

## Tabulka znaků

ET-58 používá k tisku a zobrazení odlišnou tabulku znaků než TI-58/59. Znaky se zadávají jako 2 číslice dekadického čísla v rozsahu 00 až 99 a (až na malé odchylky) odpovídají ASCII znakům sníženým o 32. Při importu programů je nutné změnit obsahy registrů **Op 01** až **Op 04**. Podrobněji viz kapitola [Tabulka znaků](#).

## Funkce HIR 20

Funkce **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).

## Organizace paměti

U původní TI-58/59 se paměť RAM dělila mezi paměť programu a paměť datových registrů. Předěl bylo možné měnit operací **Op 17**. 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).

## Přetečení negativního exponentu

U původní TI-58/59 bylo přetečení exponentu do velkých záporných hodnot indikováno jako chyba podtečení (blikající údaj 1-99). U ET-58 je přetečení velkého záporného exponentu považováno za číslo 0 a chyba není indikována.

## 6. Formát čísel

S interním formátem čísel se uživatel zpravidla nesetká. Může se s ním setkat v případě zobrazení mantisy příkazem **INV DEC**.

Čísla jsou v kalkulátoru uchována v 10 bajtech paměti. První dva bajty obsahují exponent, s vyšším bajtem na nižší adrese. Exponent je celé číslo bez znaménka, s biasem 0x8000 (32768) a s rozsahem platných hodnot 0x0001 až 0xFFFFE. Exponent s hodnotou 0x0000 je speciální případ a představuje nulu čísla (na obsahu mantisy přitom nezáleží). Druhým speciálním případem je hodnota 0xFFFF, představující nekonečno (přetečení). Po převodu na dekadický tvar čísla má exponent hodnotu v rozsahu -9863 až +9863.

Na mantisu je vyhrazeno 8 bajtů. Mantisa je binární číslo, s nejvýznamnějším bajtem na nižší adrese paměti (tj. offset 2). Nejvyšší bit mantisy (tj. bit 7 prvního bajtu na offsetu 2) má u nenulového čísla vždy hodnotu 1 a není proto v čísle vyjádřený, je skrytý. Jeho pozice je využita k indikaci znaménka čísla (1 znamená záporné číslo).

Přesnost mantisy je 19,27 dekadických číslic. Na displeji je zobrazeno max. 14 číslic, tj. 5 číslic je skrytých a slouží k zachování přesnosti výpočtů kalkulátoru. K testu přesnosti kalkulátoru lze použít populární trigonometrický test:

**9 sin cos tan INV tan INV cos INV sin**

Při správném výpočtu by mělo být výsledkem opět číslo 9. Výpočtem se rychle ztrácí přesnost a u kalkulátorů se běžně objevuje odchylka. U kalkulátoru ET-58 zůstává odchylka v rámci skrytých číslic a zobrazený výsledek bude opět číslo 9.

Více k přesnosti kalkulátorů: <http://www.datamath.org/Forensics.htm>

## 7. Klávesnice

Kalkulátor lze provozovat buď v přímém módu, kdy se kódy tlačítek ihned provádějí, nebo v programovacím módu, kdy se kódy tlačítek pouze zapisují do programu, ale neprovádí se.

Kalkulátor se ovládá sadou 45 tlačítek, rozložených v 9 řadách a 5 sloupcích. Řady jsou číslovány shora dolů, v pořadí 1 až 9. Sloupce jsou číslovány zleva doprava, s čísly 1 až 5.

Po stisku tlačítka **2nd** se použije alternativní funkce tlačítka, indikovaná číslem sloupce 6 až 10 (číslo 10 je v kódu nahrazeno číslicí 0). Po druhém stisku tlačítka **2nd** (tj. význam **3rd**) se použije druhá alternativní funkce. Je indikována sloupcem A až E.

Při zápisu programu do paměti (tlačítkem **LRN**) se do programu zapíše kód stisknutého tlačítka jako dvojice číslic, kde první číslice představuje řádek tlačítka (typicky 1 až 9) a druhá číslice sloupec tlačítka (typicky 1 až 5 pro základní funkci nebo 6 až E pro alternativní funkci).

Kódy číselných tlačítek **0** až **9** se v programu neuchovávají pomocí souřadnice tlačítka, ale jako dekadická hodnota 00 až 09.

Kromě uvedených kódů tlačítek se mohou v programu vyskytovat kódy druhotných funkcí tlačítek a speciálních příkazů, které používají kódy nedostupné přímo z klávesnice. Např. posloupnost **INV** **SBR** se do programu uloží jako příkaz **RTN** s kódem 92.

**Poznámka:** V textu příručky jsou názvy tlačítek uváděny bez případných prefixů **2nd**, které mohou být nutné k vyvolání funkce tlačítka. Např. kód tlačítka **rand** (náhodné číslo) se vyvolá stiskem tlačítek **2nd** **2nd** **CE**.

## 8. Indikátory na displeji

LCD displej obsahuje 2 řádky po 16 alfanumerických znacích. První řádek je typicky využit k zobrazení indikátorů, druhý řádek k zobrazení zadávaného čísla a výsledku operace.

Význam prvního řádku displeje lze přepínat instrukcemi:

**Op 1D** ... indikátory (implicitní zobrazení)  
**Op 1E** ... registr T  
**Op 1F** ... text (režim lze vypnout pomocí **CLR**)



### Indikátory v prvním řádku displeje:

**Deg/Rad/Grd** - indikace jednotky úhlů ve stupních, radiánech nebo gradech.  $360^\circ = 2 \cdot \pi$  radiánů = 400 gradů. Přepínač lze změnit tlačítky **Deg**, **Rad** nebo **Grad**.

**Hex/Bin/Oct** - indikuje zvolenou číselnou soustavu: dekadickou (neindikováno), hexadecimální (Hex), binární (Bin) a oktalovou (Oct). Číselnou soustavu lze volit tlačítky **DEC**, **HEX**, **BIN** a **OCT**.

**F0 až FD** - indikuje zvolené zaokrouhlování čísel 0 až 13 desetinných míst. Nastavuje se tlačítky **Fix 0** až **Fix 0D**. Posloupnost **INV Fix** (stejný význam má též **Fix 0F**) vypne zaokrouhlování zobrazeného výsledku. V tomto případě se zaokrouhlení na displeji neindikuje (nahradí se mezerami).

**EE/Eng** - indikuje způsob zobrazení exponentu. Po stisku **EE** se číslo zobrazí ve vědeckém tvaru, ve formátu mantisa a exponent. Režim lze zrušit stiskem **CLR** nebo **INV EE**. Po stisku **2nd Eng** se číslo zobrazí v technickém (inženýrském) tvaru, kdy je exponent násobkem 3. Režim Eng se vypne stiskem **INV Eng**. Jsou-li módy exponentu vypnuty,



neindikuje se na displeji nic.

**2n/3d** - indikuje první nebo druhý stisk tlačítka alternativních funkcí **2nd**. Je-li po jednom stisku **2nd** stisknuto nějaké tlačítko, provede se namísto jeho základní funkce alternativní funkce (na klávesnici zobrazena v druhém řádku). Po druhém stisku tlačítka **2nd** se na displeji objeví indikace 3d. V tom případě se provede druhá alternativní funkce zvoleného tlačítka (na klávesnici uvedena na třetím řádku). Není-li tlačítko **2nd** stisknuto, nebo je-li stisknuto 3x, není alternativní funkce aktivní, provede se základní funkce tlačítka. Tento stav není na displeji indikován (na pozici se zobrazí mezery).

**In** - indikace stisknutí tlačítka **INV**, aktivujícího inverzní funkcí. Má-li být spolu s tlačítkem **2nd** stisknuto i tlačítko **INV** (inverzní alternativní funkce), je nutné tlačítko **INV** stisknout před stiskem tlačítka **2nd**.

**Operace** - poslední pozice 1. řádku je určena k indikaci zvolené aritmetické operace: + sčítání, - odčítání, \* násobení, : dělení, & bitový AND, | bitový OR, ~ bitový XOR, \ modulo truncate (mod), / modulo floor (mod2), % procento, < posun doleva, > posun doprava, ^ mocnina, √ odmocnina, ( otevřená závorka.

## 9. Indikace chyby

Dojde-li během práce kalkulátoru k chybě, bude chyba indikována rozblíkním displeje. Současně se na začátku druhého řádku zobrazí znak **E** nebo **F**, indikující typ chyby.

Chyba typu **E** (Error) je měkká chyba. Vznikne např. dělením čísla nulou. Program může pokračovat v činnosti a chyba bude indikována až po ukončení běhu programu. Indikaci lze zrušit tlačítkem **CE** nebo **CLR**.

Chování programu při chybě lze ovlivnit pomocí přepínače 8. Je-li přepínač 8 nastaven (instrukcí **StFlg 8**), dojde k zastavení programu po vzniku měkké chyby. Není-li přepínač 8 nastaven (implicitní stav), program provede pouze nejnutnější korekci a pokračuje v chodu. Chyba bude v tom případě indikována až po ukončení programu.

Stav indikace chyby lze programově detekovat pomocí povelů **Op 18** a **Op 19**, spolu s přepínačem 7. Vyvoláním **Op 18** se nastaví přepínač 7 v případě, že chyba není indikována. V opačném případě se stav přepínače nezmění. Vyvoláním **Op 19** se nastaví přepínač 7 v případě, že chyba je indikována. V opačném případě se stav přepínače nezmění.

Třetím způsobem práce s indikací chyby je přepínač 15. Je přímo propojen s indikací chyby a lze jak detekovat (instrukcí **IfFlg 0F**), tak i nastavovat (**StFlg 0F**) či resetovat (**INV StFlg 0F**).

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

K nastavení indikace chyby lze též použít metodu používanou v programech kalkulátorů TI-58/59. Posloupností kláves **CLR 1/X** se displej rozblíká s údajem 9.9999+9999.

Chyba typu **F** (Fatal) je tvrdá chyba, neumožňující další chod programu a vedoucí k jeho okamžitému zastavení. Vznikne přetečením ukazatele programu za konec paměti, přetečením hloubky zásobníku programů či zásobníku operací, použitím neexistujícího návěští nebo použitím neplatného čísla registru.

## 10. Editor čísla

Zadávané číslo, stejně jako výsledky výpočtů, se zobrazí na 2. řádku displeje. Mantisa se zobrazí s přesností max. 14 číslic.

Před mantisou je rezervována 1 pozice pro znaménko. Na tomto místě se pro záporná čísla zobrazí '-', pro kladná čísla se ponechá mezera.

Za mantisou se zobrazí exponent (je-li aktivní režim s exponentem). Exponent je od mantisy oddělen znaménkem + nebo -. Exponent se zobrazí na 1 až 4 číslice.

Součástí mantisy může být desetinná tečka. V módu exponentu ve vědecké notaci (mantisa a exponent) se desetinná tečka zobrazí vždy za první číslicí. Není-li aktivní mód exponentu, zobrazí se desetinná tečka za číslicí jednotek. Nenásledují-li za desetinnou tečkou žádné číslice, tečka se nezobrazí.

V technickém (inženýrském) módu je v dekadickém a oktalovém módu exponent násobkem 3, před desetinnou tečkou se zobrazí 1 až 3 číslice. Je-li aktivní hexadecimální nebo binární mód zobrazení, je exponent násobkem čísla 4 a před desetinnou tečkou se zobrazí 1 až 4 číslice.

Je-li aktivní jiný režim než dekadický, zobrazí se číslice mantisy ve zvoleném číselném módu, ale exponent se vždy zobrazí jako dekadické číslo.

Tlačítko **CE** maže poslední znak mantisy nebo exponentu (podle toho kam právě probíhá zápis číslic).

Tlačítko **EE** zahájí zadávání exponentu. K zadávání mantisy se lze vrátit stiskem tlačítka tečky **.** nebo **INV EE**. Tlačítko **EE** slouží též k zahájení editace zobrazeného výsledku operace. To lze využít k odstranění skrytých číslic čísla. Jiné alternativy jsou **INV** a **Op 82**.

## 11. Číselné výrazy



Kalkulátor během výpočtů udržuje prioritu operací ve 3 stupních:

1. + sčítání, - odečítání, & bitový AND, | bitový OR, ~ bitový XOR
2. \* násobení, : dělení, \ modulo trunc (mod), / modulo floor (mod2), % procenta, < posun vlevo (násobení 2x), > posun vpravo (dělení /2)
3. ^ mocnina, V odmocnina

Při výpočtech se nejdříve vyhodnotí úroveň 3) mocnina a odmocnina, poté 2) násobení a dělení, a nakonec 1) sčítání a odčítání.

Ve výrazu lze libovolně používat závorky, a to až do 15. úrovně.

Tlačítko  slouží k záměně prvního a druhého operandu operace.

Po provedení výpočtu lze opětovným stiskem tlačítka  opakovat výpočet nejnižší úrovně. Zadáním čísla a stiskem  se zopakuje operace. Zadané číslo se použije jako první operand operace, druhý operand zůstane zachován původní.

### **Příklad:**

    [5]

  [6]

  [12]

## 12. Nepřímé adresování

Kromě přímého zadání parametrů instrukcí (jejich číselnou hodnotou) lze parametry zadat též nepřímo, pomocí datových registrů. V tom případě se jako parametr uvede číslo registru, ze kterého si kalkulátor má parametry převztít. Ke změně ukazatele na nepřímou adresu slouží tlačítko **Ind**. V některých případech se do programu uloží jiný kód instrukce (určený pro nepřímé adresování), v jiných případech se uloží kód **Ind** jako indikátor nepřímého adresování.

U instrukcí skoků obsahuje registr absolutní adresu skoku jako dekadické číslo v rozsahu 0 až 999. Kromě absolutní adresy skoku lze do registru uložit i symbol návěští. Návěští je do registru uloženo jako kód návěští, vyjádřený dekadickým číslem a vynásobený hodnotou 256 (kód návěští je ve vyšším bajtu čísla). Např. tlačítko **1/x** má hexadecimální programový kód 35. To odpovídá dekadickému číslu 53 ( $3 \cdot 16 + 5 = 53$ ). Po vynásobení číslem 256 vznikne hodnota 13568 (neboli 3500 v HEX kódu).

Samotným uvedením povelu **Ind** v programu se provede instrukce, jejíž kód je uložen v datovém registru, jehož kód je uveden za instrukcí **Ind**. Samotná instrukce **Ind** není běžně určena k provozu s vlastním parametrem, a proto je při zadávání nutné buď použít jednočíselný kód registru (číslíce 0 až 9) nebo provést krok zpět a číslo registru zpětně opravit. Kód instrukce musí být v registru uložen v dekadickém tvaru. Požaduje-li instrukce nějaké parametry, jsou načteny z části programu následující za kódem **Ind**.

### ***Příklad, nepřímé adresování registru:***

**5 STO 01** .... uloží do registru R01 hodnotu 5 (přímým adresováním)

**8 STO 05** ... uloží do registru R05 hodnotu 8 (přímým adresováním)

**RCL Ind 01** [8] ... z registru R01 přečte hodnotu 5 a použije ji jako adresu registru R05, ze kterého přečte výslednou hodnotu 8

Do programu se namísto kódu 43 01 (instrukce **RCL 01**) uloží kód 73 01 (instrukce **RCL Ind 01**).

### ***Příklad, nepřímá absolutní adresa:***

**1 2 3 STO 0 1** ... uloží hodnotu 123 do registru R01

**GTO Ind 0 1** ... z registru R01 přečte hodnotu 123 a použije ji jako skok na novou adresu

**LRN** [123 FF ...empty] ... kontrola aktuálního ukazatele programu

### ***Příklad, nepřímé návěští:***

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

**Lbi 1/x** ... zápis návěští **1/x**

**LRN** ... vypnutí programovacího módu

**RST** ... reset ukazatele zpět na 000 (pro kontrolu provedení skoku)

**1 3 5 6 8 STO 0 1** ... uložení hodnoty návěští 1/X do registru R01 (kód 1/x = 35 hex = 53 dekadicky, \* 256 = 13568)

**GTO Ind 0 1** ... z registru R01 přečte návěští a skočí na **1/x**

**LRN** (002 FF ...empty) ... kontrola cílové adresy (za návěštím **1/x**)

### ***Příklad, nepřímá instrukce:***

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

**Ind 1 2** ... nepřímá instrukce z registru R01 s parametrem '2'

**LRN** ... vypnutí programovacího módu

**4 5 STO 0 2** ... do registru R02 uloží testovací hodnotu 45

**6 7 STO 0 1** ... do registru R01 uloží dekadickou hodnotu instrukce **RCL**

**RST** ... resetuje ukazatel programu na 000

**SST** [45] ... provede instrukci na adrese 000. Z registru R01 si přečte kód instrukce **RCL** (43 hex = 67 dec), aktivuje ji, instrukce **RCL** si načte parametr následující za instrukcí **Ind**, tj. hodnotu '2'. Provede se instrukce **RCL 02**, která zobrazí obsah registru R02 (v našem případě hodnota 45).

**LRN** [003 FF ...empty] ... kontrola cílové adresy za kódy **Ind 01 02**.

## 13. Programování

Zápis posloupnosti tlačítek do programové paměti nazýváme programem. Programem se z kalkulátoru stává mocný nástroj. K dispozici je jednak uživatelský program, zapisovatelný do paměti EEPROM (tj. paměť, jejíž obsah se v procesoru uchová i po vyjmutí baterie), a jednak programová knihovna, která je uložena v paměti ROM. Programovou knihovnu nelze přepisovat editorem programu.

Režim programování se zahájí tlačítkem **LRN**. Obsah programu se zobrazuje na dvou řádcích displeje. Na spodním řádku zleva jsou 3 číslice, představující adresu aktuálního kroku programu v paměti. Adresa je v rozsahu 000 až 999 (tj. 1000 programových kroků).



Za adresou se zobrazí dvoumístný HEX kód programového bajtu na dané adrese. Za kódem bajtu je uvedena popiska tlačítka nebo funkce, která danému kódu odpovídá. Kalkulátor neumí v editoru programu rozlišit, zda se jedná o začátek instrukce nebo parametr, a proto ke všem kódům zobrazí příslušný text popisky tlačítka. Je na uživateli, aby podle kódu předešlé instrukce rozlišil, zda se jedná o kód instrukce nebo parametr.

Na horním řádku se zobrazí obsah i sousedních 4 bajtů programu. To umožní snazší orientaci v programu. Aktuální programový bajt (který je zvolen ve spodním řádku) se v horním řádku zobrazí uprostřed a je navíc orámován hranatými závorkami.

Režim programování lze aktivovat i v případě, že je aktivní některý program z knihovny (např. volbou **Pgm 02**). V tom případě se na displeji nezobrazí uživatelský program (jako je tomu u původní TI-58/59), ale obsah programu knihovny. Programem lze pouze listovat, nelze ho nijak upravovat. Program knihovny je od hlavního programu odlišen tím, že mezi adresou a kódem programového bajtu se zobrazí znak hvězdičky \* (jako indikace režimu "Read-Only", pouze čtení).

# Klávesy během programování

**SST** (Single Step) - Zvýší ukazatel programu o 1 ("další krok"). Tlačítko **SST** lze používat i během normálního (prováděcího) režimu. Po jeho stisku se provede kód instrukce, na kterou je nastaven ukazatel programu.

**BST** (Back Step) - Sníží ukazatel programu o 1 ("zpětný krok").

**Ins** (Insert) - Vloží na aktuální pozici programu prázdný bajt a odsune následující část programu. Prázdný bajt má hodnotu FF a v editoru je označen popiskou "...empty". Tento bajt má za běhu programu podobnou funkci jako instrukce **Nop** (tj. nic se neprovede), ale během editace programu má speciální význam. Tlačítka **Ins** a **Del** odsunou nebo přisunou zbylou část programové paměti, nacházející se za aktuálním ukazatelem programu. Avšak narazí-li se na kód FF, operace se přeruší. Kód FF tak působí jako jakási pružná mezera oddělující jednotlivé části programu. Zajistí, že následující úseky programu, oddělené mezerou FF, se nebudou posouvat. To je vhodné např. v případech absolutního adresování. A navíc se tím zrychlí operace **Ins** a **Del**, která může u plné paměti trvat i několik sekund. Oddělovací mezera se uplatňuje do chvíle, než úplně zmizí (vložení více bajtů). Úseky se spojí a nadále se budou posouvat společně.

*Pozor, některé instrukce umožňují zadat parametr v HEX kódu a vložit tak do kódu platný 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.*

**Del** (Delete) - Zruší bajt na aktuální pozici programu a přisune následující část programu. Během operace se uplatní prázdné bajty FF, jak bylo popsáno u instrukce **Ins**.

Je třeba zmínit, že instrukce **Ins** a **Del** nezpůsobí opravu absolutních adres skoků v programu. Z toho důvodu je vhodnější používat buď relativní skoky nebo, ještě lépe, návěští. U původního kalkulátoru TI-58/59 byly absolutní adresy upřednostňovány kvůli vyšší rychlosti, neboť vyhledání návěští v programu tam může trvat dlouhou dobu (i několik sekund). Naproti tomu u ET-58 proběhne vyhledání návěští programu velmi rychle a lze použít jako plnohodnotnou náhradu absolutních adres.

Je-li pro někoho obtížné vymýšlet, která tlačítka jako návěští může ještě



v programu použít, může pro tento účel používat číselné kódy tlačítek. Pro tento účel jsou vhodné kódy 0A0 až 0FE, které nejsou přiřazeny žádným tlačítkům a lze je dobře využít jako návěští programu. K zadání z klávesnice lze využít tlačítko **code**, následované dvěma číslicemi kódu návěští.

**LRN** (Learn) - Ukončí režim editace a navrátí kalkulátor do prováděcího režimu.

**GTO** (Go To) - Tlačítko **GTO** nelze v programovacím módu použít přímo jako ovládací tlačítko, protože se jeho kód po stisku vloží do programu. Lze ale využít v prováděcím režimu a to tak, že stiskem **LRN** přechodně vypnete programovací mód, stisknete **GTO** a zadáte 3-místnou číselnou adresu nebo tlačítko návěští, a stiskem **LRN** se vrátíte zpět do programovacího módu. Ukazatel programu bude přesunut na zadanou adresu.

**RST** (Reset) - Podobně jako **GTO** nelze použít přímo v programovacím módu, ale lze využít v prováděcím módu k převinutí ukazatele programu na adresu 000. Je-li aktivní některý knihovní program, deaktivuje se a přepne se zpět na hlavní uživatelský program.

**R/S** (Run/Stop) - Start programu nebo zastavení programu (použije se v prováděcím módu).

### **Příklad:**

**Pgm** **02** ... zvolí knihovní program 2

**GTO** **1** **2** **3** ... přesune ukazatel programu na adresu 123

**LRN** [123\*76 Lbl]... přepne do programovacího módu - kontrola adresy

**LRN** ... vypne programovací mód

**RST** ... resetuje ukazatel programu a přepne na hlavní program

**LRN** [000 FF ...empty] ... přepne na programovací mód - kontrola adresy

## 14. Tlačítka a instrukce

U každého tlačítka je uveden programový HEX kód, název tlačítka a posloupnost stisků tlačítek k jeho vyvolání.

### 00 ... 09 Základní číslice, 0...9





Symbol  ... 

Vyvolání tlačítka  ... 

Základní číslice slouží k zadání číslic v rozsahu 0 až 9, typicky se jedná o číslo v dekadickém tvaru. Používají se k zadání mantisy čísla, zadání exponentu, číslo paměťového registru, adresa absolutního skoku a jiné.

Samostatné číslice se do programu ukládají s kódem 0 až 9. Jsou-li součástí složeného čísla, jako je číslo registru nebo absolutní adresa skoku, uloží se do programu složené do BCD kódu (tj. 2 číslice na 1 bajt).

#### **Příklad:**



Posloupnost tlačítek     se do programu uloží s kódy 61 01 23.

Posloupnost tlačítek    se uloží s kódy 42 12.

### 0A ... 0F Hexadecimální číslice, 0A...0F

Symbol  ... 

Vyvolání tlačítka   ...  

Hexadecimální číslice lze použít k zadání číslic v rozsahu 10 až 15 tam, kde je to umožněno. Příkladem je zadání mantisy čísla v hexadecimálním tvaru (ne exponentu, ten je vždy zadáván v dekadickém tvaru) a zadání parametru instrukcí  a .

Zvláštností jsou parametry očekávající dekadický tvar čísla. Typicky číslo

paměťového registru. Např. instrukce **RCL 2 3** je v programu uložena ve 2 bajtech, s kódem 43 23. Během interpretace programu se číslo registru přečte složením obou číslic jako  $2 \cdot 10 + 3 = 23$ . Použije-li se jako první číslice **0A**, rozšíří se tak počet adresovatelných registrů na 110. Např. zápis **RCL 0A 3** se uloží s kódem 43 A3 a při interpretaci se použije registr  $10 \cdot 10 + 3 = 103$ .

## 10 ... 1F Písmenné návěští, A...F

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

Vyvolání tlačítka **A** ... **E**, **2nd A** ... **2nd E**, **2nd 2nd A** ... **2nd 2nd E**

Tlačítka **A** až **E** slouží k rychlému vyvolání podprogramů, typicky jedním stiskem. Podprogram se v programu označí návěští s uvedeným symbolem, např. **Lbl A**. Po stisku klávesy **A** se příslušný podprogram vyvolá.

Tlačítka jsou k dispozici ve 3 sadách. Základní sada, **A** až **E**, se aktivuje pouhým stiskem tlačítka **A** až **E**, bez prefixu **2nd**. Druhá sada, **A'** až **E'**, se aktivuje stiskem **2nd** a tlačítka **A** až **E**. Třetí sada, **A''** až **E''**, se aktivuje dvojnásobným stiskem **2nd 2nd** a tlačítka **A** až **E**.

Zvláštním případem je instrukce **F**, s programovým kódem 1F. Kód není dostupný přímo z klávesnice jako tlačítko. Lze zadat složitějším způsobem, s využitím instrukce **code**: **2nd 2nd INV 0 2nd 2nd 0F**. Takové použití by bylo nepraktické. Skutečné využití je uvnitř programu, k vyvolání podprogramu s využitím pouze 1-bajtového kódu instrukce.

Vyvolání podprogramů tlačítka **A** až **F** se provádí stejným způsobem jako vyvolání podprogramu instrukcí **SBR**. Do zásobníku adres se nejdříve uloží adresa, následující za kódem tlačítka. Poté se předá řízení podprogramu. Zásobník adres má kapacitu omezenou na 15 podprogramů.

Podprogram je ukončen instrukcí **RTN**. Ta zajistí návrat z podprogramu. Ze zásobníku adres se převezme původní adresa za instrukcí **A** až **F** a předá se na tuto adresu řízení. Pokud byl podprogram spuštěn z klávesnice, program 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.

### **Příklad:**

**RST** **LRN** ... resetuje ukazatel a aktivuje programovací mód

**Lbl** **A** ... vytvoření prozatímního návěští

**BST** **code** **1** **0F** ... krok zpět a zadání správného kódu instrukce **F**

**(** **CE** **+** **1** **)** **RTN** ... podprogram zvýší hodnotu registru **X** o 1

(instrukce **RTN** se zadá jako **INV** **SBR**)

**Lbl** **A** **code** **1** **0F** **RTN** ... testovací program zavolá podprogram **F**

**LRN** ... deaktivace programovacího módu

**1** **A** **[2]** **A** **[3]** **A** **[4]** ... zkouška, každý stisk **A** zvýší **X** o 1

## 20 Vypnutí kalkulátoru, OFF

Symbol **OFF**

Vyvolání tlačítka **2nd** **CLR**

Instrukce **OFF** zajistí vypnutí kalkulátoru a přechod do úsporného režimu. Ke stejnému vypnutí kalkulátoru dojde po určité době nečinnosti. Doba nečinnosti je nastavitelná operací **Op** **88** (implicitně 30 sekund). Opětovné zapnutí je možné stiskem tlačítka **CLR**.

Během vypnutí kalkulátoru je uchována paměť RAM, která obsahuje datové registry, pracovní registry (**X**, **Y**, **T**) a rozpracované operace. Podmínkou uchování je, aby baterie nebyla z kalkulátoru vyjmuta déle než na pár minut a aby nebylo stisknuto tlačítko **CLR** (které by kalkulátor jinak zapnulo, ale bez baterie se resetuje). Během vypnutí kalkulátoru lze

vyměnit baterii. Je-li baterie vyjmuta ze zapnutého kalkulátoru a není-li připojen externí napájecí zdroj, dojde po vložení nové baterie k resetu kalkulátoru se ztrátou obsahu RAM paměti (obsah registrů se vynuluje).

Baterie by neměla být vyjmuta z kalkulátoru během operací zápisu do paměti EEPROM - to znamená během programování uživatelského programu (např. probíhající operace **Ins**). V takovém případě může dojít k poškození obsahu paměti.

Je-li kalkulátor napájen z externího zdroje (napájecí USB konektor), zůstane displej svítit i při vypnutém kalkulátoru.

## 21 Alternativní funkce, 2nd

Symbol **2nd**

Vyvolání tlačítkem **2nd**

Tlačítko **2nd** slouží ke změně významu následujícího tlačítka na alternativní funkci. Většina tlačítek má 2 alternativní funkce. Po jednom stisku tlačítka **2nd** se provede první alternativní funkce, po dvojnásobném stisku **2nd 2nd** se provede druhá alternativní funkce. Trojnásobným stiskem **2nd 2nd 2nd** se navrátí zpět k základním funkcím.

Kód tlačítka **2nd** se do programu neukládá. Vždy se ukládá alternativní kód následujícího tlačítka.

### **Příklad:**

**2** **ln x** [.6931...] ... vypočte přirozený logaritmus čísla

**2** **2nd** **ln x** [.3010...] ... dekadický logaritmus čísla (instrukce **log**)

**2** **2nd** **2nd** **ln x** [1] ... binární logaritmus čísla (instrukce **log2**)

## 22 Inverze funkce, INV

Symbol **INV**

Vyvolání tlačítkem **INV**

Tlačítko **INV**, stisknuté před jiným kódem tlačítka, u mnoha tlačítek vyvolá jejich obrácenou funkci. V některých případech jde o doplňkovou alternativní funkci.

Kód tlačítka se do programu ukládá jako bajt 22. Většina instrukcí prefix **INV** buď přímo obslouží nebo alespoň resetuje. V případě návěští **Lbl** se stav prefixu **INV** zachová. To lze využít k rozlišení funkce programu uvedením instrukce **INV** před návěštěm programu.

### **Příklad:**

**RST** **LRN** ... aktivuje programovací mód

**Lbl** **B** ... návěští prvního podprogramu

**INV** ... následující instrukce za **Lbl** se bude invertovat

**Lbl** **A** ... návěští druhého podprogramu

**sin** ... výpočet sinus

**RTN** ... návrat z podprogramu (zadáno tlačítky **INV** **SBR**)

**LRN** ... návrat do prováděcího módu

**3** **0** **A** [0.5] **B** [30] ... Podprogram **A** vypočte sinus zadaného úhlu, podprogram **B** přepočte hodnotu zpět na úhel.

## 23 Přirozený logaritmus a exponent, $\ln x$

Symbol  **$\ln x$**

Vyvolání tlačítkem  **$\ln x$**

Tlačítkem  **$\ln x$**  se vypočte přirozený logaritmus čísla na displeji. Přirozený logaritmus používá jako základ Eulerovu konstantu s hodnotou 2.718281828459. Stiskne-li se nejdříve tlačítko **INV**, provede se inverzní funkce, přirozený exponent.

Argumentem funkce **ln x** musí být kladné, nenulové číslo. V případě nuly se displej rozbliká s hodnotou 9.9999+9999, jako příznak chyby. Pro záporné číslo se provede výpočet absolutní hodnoty čísla a displej se opět rozbliká s indikací chyby.

Argumentem funkce **INV ln x** může být jak kladné, tak i záporné číslo, v rozsahu zhruba -23025 až +23025. Číslo mimo tento rozsah způsobí přetečení údaje a indikaci chyby.

### **Příklad:**

**5 ln x** [1.6094...] ... vypočte přirozený logaritmus čísla 5 (tj. 1.6094...)

**INV ln x** [5] ... vypočte přirozený exponent čísla na displeji (tj. 5)

## 24 Oprava chyby, CE

Symbol **CE**

Vyvolání tlačítkem **CE**

Tlačítkem **CE** lze zrušit indikaci chyby E, projevující se blikáním displeje. Spolu s nulováním indikace chyby je provedena korekce pracovních registrů X, T a Y tak, aby neobsahovaly číslo s indikací přetečení - tj. indikované číslo 9.9999+9999 změní na hodnotu zhruba 7.0773+9863, což je maximální zobrazitelná hodnota kalkulátoru.

Během editace čísla na displeji je tlačítkem **CE** smazán poslední znak zadávaného údaje. Probíhá-li editace mantisy, je smazán poslední znak mantisy. Probíhá-li editace exponentu, je smazán poslední znak exponentu. Je-li mazán exponent s hodnotou 0, exponent se zruší a přejde se na editaci mantisy.

## 25 Vymazání displeje, CLR

Symbol **CLR**

Vyvolání tlačítkem **CLR**

Tlačítko **CLR** provede několik inicializačních operací. Resetuje započaté aritmetické operace, vynuluje indikaci chyby, vypne textový mód displeje (aktivovaný příkazem **Op 1F**), navrátí implicitní fonty displeje, vypne mód exponentu **EE**, vynuluje registr X a zahájí editaci nového čísla s výchozí hodnotou 0.

Tlačítko nenuluje registr T, datové registry ani HIR registry.

Speciální funkcí tlačítka je zapnutí kalkulátoru z vypnutého stavu. Tlačítko slouží též k vypnutí kalkulátoru, a to v případě, že je před jeho stiskem stisknuta prefixová klávesa **2nd** (funkce **OFF**).

## 26 Podprogram s nepřímou adresou, SBR Ind

Symbol **SBR Ind**

Vyvolání tlačítka **SBR 2nd y^x**

Za příkazem **SBR Ind** následuje jako parametr číslo registru, obsahující adresu skoku nebo symbol návěští. Příkaz zavolá podprogram s adresou nebo návěštím, obsaženým v registru. Způsob adresování byl uveden v kapitole [Nepřímé adresování](#).

Jinak příkaz funguje stejně jako příkaz **SBR** (viz [71 Podprogram, SBR](#)), tj. do zásobníku programu uloží aktuální adresu (max. hloubka zásobníku je 15 adres) a po ukončení podprogramu (ukončení instrukcí **RTN** [92 Návrat z podprogramu, RTN](#)) pokračuje na původní uschované adrese.

## 27 Nepřímá interní instrukce, HIR Ind

Symbol **HIR Ind**

Vyvolání tlačítka **2nd INV 2nd y^x** (pouze v programu)

Instrukce **HIR Ind** funguje obdobně jako instrukce **HIR**, ovšem namísto pracovního HIR registru použije datový registr, jehož index je obsažen v HIR registru, uvedeném jako parametr instrukce.



*Poznámka: Z interních důvodů nelze funkci **HIR** **Ind** vyvolat přímo z klávesnice posloupností **HIR** **Ind**. Funkce může být takto použita pouze uvnitř programu. Z klávesnice je možné vyvolání funkce zadáním jejího kódu pomocí povelu **code** (tj. stiskem **2nd** **2nd** **INV** **27**).*

### **Příklad:**

**4** **5** **STO** **01** ... uložení testovacího vzoru 45 do datového registru R01

**1** **HIR** **05** ... uložení hodnoty 1 do HIR registru H5

**code** **27** **15** [45] ... Zavolání funkce **HIR** **Ind** kódem 27. Funkce načte z registru H5 hodnotu 1 a z registru R01 přečte obsah 45.

## 28 Dekadický logaritmus a exponent, log

Symbol **log**

Vyvolání tlačítka **2nd** **ln x**

Tlačítkem **log** se vypočte dekadický logaritmus čísla na displeji. Dekadický logaritmus používá jako základ číslo 10. Stiskne-li se nejdříve tlačítko **INV**, provede se inverzní funkce, dekadický exponent.

Argumentem funkce **log** musí být kladné, nenulové číslo. V případě nuly se displej rozblíká s hodnotou 9.9999+9999, jako indikace chyby. Pro záporné číslo se provede výpočet absolutní hodnoty čísla a displej se opět rozblíká s indikací chyby.

Argumentem funkce **INV** **log** může být jak kladné, tak i záporné číslo, v rozsahu zhruba -9863 až +9863. Číslo mimo tento rozsah způsobí přetečení údaje a indikaci chyby.

### **Příklad:**

**5** **log** [.69897...] ... dekadický logaritmus čísla 5 (tj. 0.69897...)

**INV** **log** [5] ... dekadický exponent čísla na displeji (tj. 5)

## 29 Vymazání programu a registru T, CP

Symbol **CP**

Vyvolání tlačítka **2nd** **CE**

Tlačítko **CP** má dvě funkce - vymazání uživatelského programu a vymazání registru T.

Je-li tlačítko **CP** vyvoláno z klávesnice, provede se vymazání uživatelského programu. Před vymazáním se zobrazí dotaz k potvrzení operace. Stiskem tlačítka **1** se operace potvrdí a uživatelský program se vymaže. Registr T se vymaže i v případě zamítnutí operace vymazání programu. V takovém případě slouží příkaz pouze k vynulování registru T.

Je-li tlačítko **CP** vyvoláno z programu, provede se pouze vynulování registru T.

## 2B Zadání kódu instrukce, code

Symbol **code**

Vyvolání tlačítka **2nd** **2nd** **INV**

Instrukcí **code** lze zadat přímý číselný kód tlačítka. Umožňuje to zadat takové kódy tlačítek, které jsou buď špatně nebo vůbec dostupné z klávesnice. Jako parametr instrukce se zadá dvoumístný kód tlačítka. Kód tlačítka se provede stejně, jako by tlačítko bylo stisknuto na klávesnici. Touto cestou lze zadat nestandardní kódy tlačítek i do programu, během programování. Samotný kód instrukce 2B se za běhu programu ignoruje.

### **Příklad:**

**5** **STO** **01** ... uložení testovací hodnoty 5 do registru R01

**CLR** [0] ... vymazání displeje

**code** **43** **01** [5] ... zavolání kódu tlačítka **RCL** a provedení instrukce **RCL** **01** (navrátí obsah 5)

## 2C Dvojkový logaritmus a exponent, log2

Symbol **log2**

Vyvolání tlačítka **2nd** **2nd** **ln x**

Tlačítkem **log2** se vypočte dvojkový logaritmus čísla na displeji. Dvojkový logaritmus používá jako základ číslo 2. Stiskne-li se nejdříve tlačítko **INV**, provede se inverzní funkce, dvojkový exponent.

Argumentem funkce **log2** musí být kladné, nenulové číslo. V případě nuly se displej rozblíká s hodnotou 9.9999+9999, jako indikace chyby. Pro záporné číslo se provede výpočet absolutní hodnoty čísla a displej se opět rozblíká s indikací chyby.

Argumentem funkce **INV** **log2** může být jak kladné, tak i záporné číslo, v rozsahu -32767 až +32767. Číslo mimo tento rozsah způsobí přetečení údaje a indikaci chyby.

### **Příklad:**

**5** **log2** [2.3219...] ... vypočte dvojkový logaritmus čísla 5 (tj. 2.3219...)

**INV** **log2** [5] ... vypočte dvojkový exponent čísla na displeji (tj. 5)

## 2D Generátor náhodného čísla, rand

Symbol **rand**

Vyvolání tlačítka **2nd** **2nd** **CE**

Funkce **rand** vypočte náhodné číslo v rozsahu 0 až 1 (nebo po zadané hraniční číslo, s prefixem **INV**), včetně hodnoty 0, ale vyjma hodnoty 1. K výpočtu náhodného čísla se používá LCG generátor (Linear Congruential Generator, Lineární kongruentní generátor) a vzorec  $\text{RandSeed} = \text{RandSeed} * 214013 + 2531011$ . Seed generátoru má rozsah 32 bitů. Vygenerované číslo je převedeno na float vydělením hodnotou  $2^{32}$ . To zajistí, že výsledné náhodné číslo je v rozsahu 0 až 1, včetně nuly, ale

vyjma hodnoty 1. Omezený počet číslic výchozího čísla (9 a půl číslice, rozsah čísla 0 až 4294967295) zajišťuje takovou granularitu údaje, že ani po vynásobení nedojde k přetečení na hraniční koncovou hodnotu 1.

Je-li před instrukcí **rand** stisknut prefix **INV**, je vygenerované náhodné číslo vynásobeno hodnotou registru X. To umožní generovat čísla v daném rozsahu, od nuly po zadané maximální číslo, ovšem vyjma zadaného maximálního čísla.

Generátor náhodného čísla při každém použití počítá neustále dál a to zajistí, že vygenerované sekvence čísel se neopakují. Navíc je při každém resetu kalkulátoru seed generátoru načteno z EEPROM a uložena nová hodnota. To zajistí, že se generované sekvence neopakují ani po resetu kalkulátoru (resp. po vyjmutí baterie).

Neopakováním je myšlena posloupnost malých vygenerovaných čísel. Budou-li generována velká čísla pokrývající rozsah generátoru (9.5 číslic), dojde po delší době k opakování generované sekvence.

K řízení seed generátoru slouží **Op 51** a **Op 52**.

### ***Příklad programu pro hod kostkou:***

**Lbl A ( 6 INV rand Int + 1 ) RTN** ... stisk **A** generuje čísla 1 až 6

## **30 Tangens, tan**

Symbol **tan**

Vyvolání tlačítka **2nd 1/x**

Funkce **tan** vypočte tangens úhlu. Úhel je zadán v jednotkách nastavených přepínači **Deg**, **Rad** nebo **Grad**. Je-li k dispozici úhel vypočtený v radiánech, lze ho převést na aktuální úhlovou míru funkcí **Op 73**.

Zadáním prefixu **INV** před instrukcí **tan** se provede opačná funkce - arkus tangens. Výsledkem je úhel v aktuálně nastavené úhlové míře. Je-li

potřeba převést výsledek na radiány, lze k tomu použít funkci **Op** **72**.

### **Příklad:**

**5** **0** **tan** [1.19175...] ... tangens úhlu 50° je hodnota 1.19175...

**9** **9** **9** **9** **9** **9** **9** **9** **INV** **tan** [90] ... arkus tangens velkého čísla je 90°

### **Poznámka:**

*Výpočty arkus tangens úhlů pohybujících se kolem hodnot +90° a -90° se již dostávají mimo rozsahy kalkulatoru, a tak může mít výsledek sníženou přesnost. Např. v uvedeném příkladu 99999999 **INV** **tan** se zobrazí výsledek 90°, přestože správná hodnota by měla být zhruba 89.999999427°.*

## 31 Programování, LRN

Symbol **LRN**

Vyvolání tlačítkem **LRN**

Tlačítko **LRN** aktivuje nebo deaktivuje režim programování. Režim programování byl blíže popsán v sekci [Programování](#).

## 32 Záměna registrů X a T, x<>t

Symbol **x<>t**

Vyvolání tlačítkem **x<>t**

Tlačítkem **x<>t** je možné zaměnit registry X a T. Registr X je pracovní registr neboli též obsah displeje. Registr T je pomocný registr (temporary). Slouží k porovnávání čísel, k převodu polárních a kartézských souřadnic a při výpočtu komplexních čísel.

Registr X je nulován tlačítky **CLR** a **CE**. Registr T je nulován tlačítkem **CP**

(pozor, slouží též k vymazání obsahu uživatelského programu).

Registr T není běžně viditelný na displeji. Pomocí instrukce **Op 1E** je možné zapnout speciální mód displeje, kdy je na horním řádku displeje zobrazen registr T. To může být výhodné např. při výpočtu komplexních čísel, k současnému zobrazení reálné a imaginární složky čísla. Zpětné přepnutí na standardní formát displeje s řádkem přepínačů je možné instrukcí **Op 1D**.

### 33 Druhá mocnina čísla, $x^2$

Symbol  **$x^2$**

Vyvolání tlačítkem  **$x^2$**

Tlačítkem  **$x^2$**  se vypočte druhá mocnina čísla, nebo-li násobek čísla samo se sebou.

### 34 Druhá odmocnina čísla, $\sqrt{x}$

Symbol  **$\sqrt{x}$**

Vyvolání tlačítkem  **$\sqrt{x}$**

Tlačítkem  **$\sqrt{x}$**  se vypočte druhá odmocnina čísla. Číslo nesmí být záporné. Je-li proveden výpočet záporného čísla, vypočte se odmocnina absolutní hodnoty čísla a nastaví se indikace chyby E (displej bliká).

### 35 Převrácená hodnota čísla, $1/x$

Symbol  **$1/x$**

Vyvolání tlačítkem  **$1/x$**

Tlačítkem  **$1/x$**  se vypočte převrácená hodnota čísla. Je-li číslem nula, zobrazí se hodnota 9.9999+9999 a nastaví se indikace chyby E (displej bliká). To se často používá v programech k zapnutí indikace chyby a

k indikaci chybného běhu programu.

## 36 Výběr programu knihovny, Pgm

Symbol **Pgm**

Vyvolání tlačítka **2nd** **LRN**

Tlačítkem **Pgm** lze vybrat program z knihovny. Jako parametr se uvede dvouciferné číslo zvoleného programu, počínaje číslem 01. Číslem 00 se aktivuje hlavní uživatelský program. Po volbě programu se na displeji na krátkou chvíli zobrazí název knihovny, číslo zvoleného programu a délka programu v bajtech. Počet programů v knihovně je 50. Při volbě programu mimo rozsah se rozblíká indikace chyby E.

Je-li zvolen některý program z knihovny, budou se nadále spouštět podprogramy z tohoto programu.

Po aktivaci režimu programování tlačítkem **LRN** se zobrazí obsah vybraného knihovního programu. Programem lze listovat a prohlížet jej, nelze ho však upravovat. Knihovní program je označen znakem hvězdičky mezi údajem adresy a aktuálního bajtu (příznak read-only, pouze pro čtení).

Volbou programu 00 se vybere uživatelský program (na displeji se na chvíli zobrazí informace "Main Program"). Pouze uživatelský program je možné upravovat. Podobnou funkci má i tlačítko **RST**, které také přepne výběr na uživatelský program.

Uvedením prefixu **INV** před tlačítkem **Pgm** se pracovní registr X (tj. obsah displeje) nastaví na hodnotu čísla vybraného knihovního programu a na displeji krátce probliknou údaje o vybraném programu, podobně jako po přepnutí programu. Tímto způsobem je možné programově testovat, který program je právě aktivní.

Je-li funkce **Pgm** použita uvnitř běžícího programu, nedojde k trvalému přepnutí programu. Přepnutí se uplatní pouze pro následující instrukci vyvolání podprogramu (písmeno **A** až **F** nebo podprogram **SBR**). Tímto způsobem lze zavolat program z jiného knihovního programu, či dokonce z hlavního uživatelského programu.

Uvedením kódu **Ind** lze funkci změnit na nepřímé adresování **Pgm Ind** (viz [62 Nepřímý výběr programu knihovny, Pgm Ind](#)).

## 37 Převod kartézských a polárních souřadnic, P->R

Symbol **P->R**

Vyvolání tlačítka **2nd** **x<>t**

Tlačítkem **P->R** se převedou souřadnice z polárního vyjádření na souřadnice kartézské. Před operací obsahuje registr T (tj. pomocný registr) radius (poloměr) a registr X (obsah displeje) obsahuje úhel. Úhel se udává v aktuálně vybrané úhlové míře (tlačítka **Deg**, **Rad** a **Grad**). Po operaci obsahuje registr T (pomocný registr) souřadnici X, registr X (obsah displeje) obsahuje souřadnici Y. Úhel lze před operací převést z radiánů na aktuálně vybranou úhlovou míru instrukcí **Op 73**.

Uvedením prefixu **INV** před instrukcí **P->R** se provede opačná operace, převod kartézských souřadnic na polární. Před operací obsahuje registr T souřadnici X, registr X obsahuje souřadnici Y. Po operaci obsahuje registr T radius (poloměr) a registr X obsahuje úhel. Úhel je udán v aktuálně vybrané úhlové míře. Je-li potřeba po operaci převést úhel na radiány, lze to provést instrukcí **Op 72**.

Instrukcí **Op 1E** lze aktivovat speciální režim displeje, při kterém se na prvním řádku displeje zobrazí obsah registru T. To může usnadnit použití převodu souřadnic. K obnovení módu displeje slouží instrukce **Op 1D**.

### **Příklad.**

**1 0** **x<>t** ... zadání radiusu 10 do registru T

**3 0** ... zadání úhlu 30° do registru X

**P->R** [5] ... převod polárních souřadnic na kartézské. Na displeji se zobrazí souřadnice Y = 5

**x<>t** [8.6602...] ... přepnutím X a T se zobrazí souřadnice X = 8.6602...

**x<>t** [5] ... přepnutí registrů X a T zpět



**INV** **P->R** [30] ... zpětný přepoččet, zobrazí se úhel 30°

**x<>t** [10] ... zobrazení registru T s radiusem 10

## 38 Sinus, sin

Symbol **sin**

Vyvolání tlačítky **2nd** **x^2**

Funkce **sin** vypočte sinus úhlu. Úhel je zadán v jednotkách nastavených přepínači **Deg**, **Rad** nebo **Grad**. Je-li k dispozici úhel vypočtený v radiánech, lze ho převést na aktuální úhlovou míru funkcí **Op** **73**.

Zadáním prefixu **INV** před instrukcí **sin** se provede opačná funkce - arkus sinus. Výsledkem je úhel v aktuálně nastavené úhlové míře. Je-li potřeba převést výsledek na radiány, lze k tomu použít funkci **Op** **72**.

Úhel vypočtený funkcí arkus sinus je v rozsahu -90° až +90°. Vstupní hodnota funkce arkus sinus musí být v rozsahu -1 až +1. Leží-li mimo uvedený rozsah, údaj se omezí do platného rozsahu a indikuje se chyba E (displej bliká).

## 39 Kosinus, cos

Symbol **cos**

Vyvolání tlačítky **2nd** **Vx**

Funkce **cos** vypočte kosinus úhlu. Úhel je zadán v jednotkách nastavených přepínači **Deg**, **Rad** nebo **Grad**. Je-li k dispozici úhel vypočtený v radiánech, lze ho převést na aktuální úhlovou míru funkcí **Op** **73**.


Zadáním prefixu **INV** před instrukcí **cos** se provede opačná funkce - arcus kosinus. Výsledkem je úhel v aktuálně nastavené úhlové míře. Je-li potřeba převést výsledek na radiány, lze k tomu použít funkci **Op** **72**.



Úhel vypočtený funkcí arkus kosinus je v rozsahu 0° až +180°. Vstupní hodnota funkce arkus kosinus musí být v rozsahu -1 až +1. Leží-li mimo uvedený rozsah, údaj se omezí do platného rozsahu a indikuje se chyba E (displej bliká).

### 3A Teplota, Temp

Symbol 

Vyvolání tlačítky   


Tlačítkem  lze zjistit teplotu čipu procesoru, a tím i teplotu v místnosti (procesor se díky nízké frekvenci prakticky nezahřívá). Rozlišení měření teploty je 1° C. Údaj teploty je velmi nepřesný, i po kalibraci se může lišit o několik stupňů, a je pouze orientační.

Měření teploty je nutné před prvním použitím nejdříve zkalibrovat. Bez kalibrace se zobrazí údaj např. 100. Kalibraci provedete tak, že do kalkulačky vepíšete aktuální teplotu místnosti v celých stupních, např. 24. Stiskem   se nastavený údaj uloží do EEPROM paměti jako rozdíl mezi skutečnou a naměřenou teplotou. Od té doby se bude k naměřenému údaji přičítat rozdíl uložený v EEPROM paměti. Zpravidla tak lze dosáhnout odchylku měření teploty do 2°C, ovšem pouze u teplot blízkých teplotě při kalibraci. U vzdálenějších teplot může být odchylka větší. Kalibraci teploty lze stejným postupem kdykoliv opakovat.

### 3B Záměna registrů X a Y, x<>y

Symbol 

Vyvolání tlačítky   

Během zadávání aritmetických operací obsahuje displej hodnotu pracovního registru X. Je-li výpočet prováděn se dvěma registry (např. instrukce pro sčítání), obsahuje zásobník aritmetických operací druhý registr, Y. Tlačítkem  lze registry operace zaměnit. To může být potřebné u operací s důležitostí pořadí operandů, jako je např. dělení či odečítání.

Tlačítko **x<>y** se může uplatnit též při opakování naposledy prováděného výpočtu tlačítkem **=**. V tom případě je druhý zadáný parametr uložen v registru Y, do registru X (údaj na displeji) se zadává první operand operace.

### **Příklad:**

**3** **y^x** **2** **=** [9] ... vypočte hodnotu  $3^2 = 9$

**5** **=** [25] ... vypočte hodnotu  $5^2 = 25$

**0** **.** **5** **x<>y** **2** ... do registru Y uloží hodnotu 0.5

**9** **=** [3] ... vypočte hodnotu  $9^{0.5} = 3$

## 3C Hyperbolický sinus, sinh

Symbol **sinh**

Vyvolání tlačítky **2nd** **2nd** **x^2**

Tlačítko **sinh** slouží k výpočtu funkce hyperbolický sinus.

Uvedením prefixu **INV** před instrukcí **sinh** se provede inverzní operace, hyperbolický arkus sinus.

### **Příklad:**

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

**INV** **sinh** [1] ... zpětný výpočet  $\operatorname{asinh}(1.1752...) = 1$

## 3D Hyperbolický kosinus, cosh

Symbol **cosh**

Vyvolání tlačítky **2nd** **2nd** **Vx**

Tlačítko **cosh** slouží k výpočtu funkce hyperbolický kosinus.

Uvedením prefixu **INV** před instrukcí **cosh** se provede inverzní operace, hyperbolický arkus kosinus.

**Příklad:**

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

**INV** **cosh** [1] ... zpětný výpočet  $\operatorname{acosh}(1.5430...) = 1$

### 3E Hyperbolický tangens, tanh

Symbol **tanh**

Vyvolání tlačítka **2nd** **2nd** **1/x**

Tlačítko **tanh** slouží k výpočtu funkce hyperbolický tangens.

Uvedením prefixu **INV** před instrukcí **tanh** se provede inverzní operace, hyperbolický arkus tangens.

**Příklad:**

**1** **tanh** [0.76159...] ... výpočet  $\tanh(1) = 0.76159...$

**INV** **tanh** [1] ... zpětný výpočet  $\operatorname{atanh}(0.76159...) = 1$

### 40 Nepřímé adresování, Ind

Symbol **Ind**


Vyvolání tlačítka **2nd** **Ind**

Tlačítko **Ind** (Indirect) slouží k zadání nepřímých parametrů operace, kdy potřebný údaj není převzat z kódu instrukce, ale z obsahu datového registru. Bližší informace byly popsány v kapitole [Nepřímé adresování](#).

## 41 Krok programu vpřed, SST

Symbol 

Vyvolání tlačítkem 


Tlačítkem  (Single Step) se v režimu programování zvýší ukazatel adresy programu o 1.


V prováděcím režimu se provede 1 instrukce programu, čímž je možné program krokovat z důvodu ladění. Bude-li kombinováno krokování programu se spouštěním podprogramů, nemusí docházet ke správnému navracení z podprogramů (kalkulátor si nebude pamatovat návratovou adresu z podprogramu).






Bližší informace naleznete v kapitole [Programování](#).

## 42 Uložení čísla do datového registru, STO

Symbol 

Vyvolání tlačítkem 

Pomocí tlačítka  (Store) lze uložit číslo na displeji do datového registru. Jako parametr instrukce se zadává dvoumístný kód čísla registru, od 00 do 99.

Uvedením kódu  za instrukcí , ale ještě před zadáním parametru, se instrukce  změní na nepřímou instrukci   (s kódem 72, viz [72 Nepřímé uložení čísla do registru, STO Ind](#)). Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

## 43 Vyvolání čísla z datového registru, RCL

Symbol 

Vyvolání tlačítkem 

Pomocí tlačítka **RCL** (Recall) lze vyvolat číslo z datového registru na displej. Jako parametr instrukce se zadává dvoumístný kód čísla registru, od 00 do 99.

Uvedením kódu **Ind** za instrukcí **RCL**, ale ještě před zadáním parametru, se instrukce **RCL** změní na nepřímou instrukci **RCL Ind** (s kódem 73, viz [73 Nepřímé vyvolání obsahu registru, RCL Ind](#)). Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

## 44 Přičtení a odečtení čísla od datového registru, SUM

Symbol **SUM**

Vyvolání tlačítkem **SUM**

Pomocí tlačítka **SUM** (Summation) lze přičíst číslo na displeji k datovému registru. Jako parametr instrukce se zadává dvoumístný kód čísla registru, od 00 do 99.

Uvedením kódu **Ind** za instrukcí **SUM**, ale ještě před zadáním parametru, se instrukce **SUM** změní na nepřímou instrukci **SUM Ind** (s kódem 74, viz [74 Nepřímé přičtení a odečtení čísla z registru, SUM Ind](#)). Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

Uvedením prefixu **INV** před instrukcí **SUM** se provede opačná funkce - odečtení čísla od datového registru.

## 45 Mocnina a odmocnina, y^x

Symbol **y^x**





Vyvolání tlačítkem **y^x**

Instrukce **y^x** umocní číslo Y (první operand, v zásobníku operací) číslem X (druhý operand, číslo na displeji). Stiskne-li se nejdříve prefix **INV**, provede se inverzní operace, odmocnina.

Prvním operandem Y by běžně mělo být nezáporné číslo. Pouze v případě,

kdy je mocnitelem X celé číslo, může být prvním operandem záporné číslo. V tom případě je výsledek záporný, pokud byl mocnitel liché číslo.


### **Příklad:**

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

## 46 Vložení prázdného bajtu do programu, Ins

Symbol 

Vyvolání tlačítka  

Tlačítko  (Insert), stisknuté v režimu programování, vloží na aktuální pozici programu prázdný bajt s hodnotou FF. Následující data se odsunou, až po konec paměti nebo po další prázdný bajt FF. Absolutní adresy se při přesunu nepřepočítávají.


Bližší informace naleznete v kapitole [Programování](#).


*Pozor, některé instrukce umožňují zadat parametr v HEX kódu a vložit tak do kódu platný 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.*

## 47 Vymazání datových registrů, CMs

Symbol 

Vyvolání tlačítka  

Tlačítkem  lze vynulovat obsahy všech datových registrů (neuplatní se na HIR registry).

Je-li před vyvoláním funkce stisknut prefix , provede se pouze vymazání datových registrů R01 až R06 (registry použité pro statistické funkce), registr T a registr X.

## 48 Záměna čísla s datovým registrem, Exc

Symbol **Exc**

Vyvolání tlačítka **2nd RCL**

Pomocí tlačítka **Exc** (Exchange) lze zaměnit číslo na displeji s obsahem datového registru. Jako parametr instrukce se zadává dvoumístný kód čísla registru, od 00 do 99.

Uvedením kódu **Ind** za instrukcí **Exc**, ale ještě před zadáním parametru, se instrukce **Exc** změní na nepřímou instrukci **Exc Ind** (s kódem 63, viz [63 Nepřímá záměna čísla s datovým registrem, Exc Ind](#)). Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

## 49 Vynásobení a vydělení datového registru, Prd

Symbol **Prd**

Vyvolání tlačítkem **2nd SUM**

Pomocí tlačítka **Prd** (Product) lze vynásobit datový registr číslem na displeji. Jako parametr instrukce se zadává dvoumístný kód čísla registru, od 00 do 99.

Uvedením kódu **Ind** za instrukcí **Prd**, ale ještě před zadáním parametru, se instrukce **Prd** změní na nepřímou instrukci **Prd Ind** (s kódem 64, viz [64 Nepřímé násobení a dělení registru, Prd Ind](#)). Bližší informace k nepřímému adresování naleznete v kapitole [Nepřímé adresování](#).

Uvedením prefixu **INV** před instrukcí **Prd** se provede opačná funkce - vydělení datového registru číslem na displeji.

## 4A Zjištění napětí baterie, Bat

Symbol **Bat**



Vyvolání tlačítka **2nd** **2nd** **SST**

Tlačítkem **Bat** lze zjistit napětí napájecí baterie. Napětí se zobrazí ve voltech. Z napětí baterie lze usuzovat stav vybití baterie. Čerstvá baterie CR2032 má napětí kolem 3V. Kalkulátor je schopen pracovat od minimálního napětí asi 2.5V. Při napájení z USB konektoru bude indikované napájecí napětí kolem 2.9V (používá se stabilizátor, který napětí sníží).

Uvedením prefixu **INV** před stiskem **Bat** se údaj napětí baterie zobrazí v procentech. Údaj 0 až 100% odpovídá napětí baterie 2.4V až 2.9V.

Přesnost údaje měření napětí baterie není zaručená, slouží spíše pouze k orientačním účelům.

## 4B Faktoriál, x!

Symbol **x!**

Vyvolání tlačítka **2nd** **2nd** **STO**

Tlačítko **x!** slouží k výpočtu faktoriálu. Faktoriál je číslo, které vznikne vynásobením hodnot 1, 2, 3, ... až po zadané vstupní číslo x. Kalkulátor počítá faktoriál pomocí aproximační funkce (Stieltjes Gamma function). To umožňuje rychlý a přesný výpočet, včetně faktoriálů desetinných čísel.

Vstupní číslo nesmí být záporné. Maximální hodnota faktoriálu, kterou kalkulátor umí vypočítat, je 3208 (výsledek 8.61680144+9856).

Uvedením prefixu **INV** před stiskem **x!** se namísto výpočtu aproximační funkcí použije opakované celočíselné násobení. Vstupem může být celé číslo v rozsahu 0 až 3209. Tato metoda není přesnější ani rychlejší než výpočet pomocí aproximační funkce, slouží pouze jako referenční hodnota.

### **Příklad:**

**6** **x!** [720] ... faktoriál čísla 6! =  $1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720$

**INV** **6** **x!** [720] ... referenční faktoriál násobením 6! = 720

**6** **.** **1** **x!** [868.9568...] ... desetinný faktoriál  $6.1! = 868.9568...$

## 4C Přirozený logaritmus faktoriálu, $\ln x!$

Symbol **ln x!**

Vyvolání tlačítka **2nd** **2nd** **RCL**

Tlačítko **ln x!** počítá přirozený logaritmus faktoriálu. Vstupní hodnotou je nezáporné číslo, včetně čísel desetinných. Na rozdíl od funkce pro výpočet faktoriálu **n!**, tato funkce není tak omezena rozsahem kalkulátoru. Vstupem může být prakticky jakékoliv kladné číslo.

### **Příklad:**

**6** **ln x!** [6.57925...] ... přirozený logaritmus  $\ln 6! = 6.57925...$

**6** **n!** **ln x** [6.57925...] ... kontrolní výpočet

## 4D Dekadický logaritmus faktoriálu, $\log x!$

Symbol **log x!**

Vyvolání tlačítka **2nd** **2nd** **SUM**

Tlačítko **log x!** počítá dekadický logaritmus faktoriálu. Vstupní hodnotou je nezáporné číslo, včetně čísel desetinných. Na rozdíl od funkce pro výpočet faktoriálu **n!**, tato funkce není tak omezena rozsahem kalkulátoru. Vstupem může být prakticky jakékoliv kladné číslo. Funkci lze využít i k počítání velmi velkých faktoriálů.

### **Příklad 1:**

**6** **log x!** [2.85733...] ... dekadický logaritmus  $\log 6! = 2.85733...$

**6** **n!** **log x** [2.85733...] ... kontrolní výpočet

## Příklad 2, 123456789!:

**1** **2** **3** **4** **5** **6** **7** **8** **9** **log x!** [945335859.45538] ... logaritmus faktoriálu

**INV** **Int** **INV** **log** [2.8535...] ... exponent desetinné části

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

*Poznámka: Podle kalkulátoru je výsledek  $123456789! = 2.853512521$  **7299** \*  $10^{945335859}$ . Správná hodnota by měla být  $2.853512521$  **9128** \*  $10^{945335859}$ . Mantisa výsledku se shoduje na 10 číslic. To je dáno použitím logaritmu. Vnitřní přesnost kalkulátoru je 19 číslic. S takovou přesností kalkulátor vypočítá logaritmus faktoriálu. Celočíselná část logaritmu představuje 9 číslic exponentu ( $10^{945335859}$ ). Po odstranění 9 číslic celočíselné části zůstane mantisa s přesností 10 číslic, což je dosažitelná přesnost mantisy výsledku. Při rozdělení logaritmu na část exponentu a mantisy je nutné počítat se snížením přesnosti mantisy výsledku.*

## 4E Modulo floor, mod2

Symbol **mod2**

Vyvolání tlačítka **2nd** **2nd** **y^x**

Operace modulo vydělí první operand Y (v zásobníku) druhým operandem X (na displeji), převede výsledek na celé číslo, vynásobí jím druhý operand X a odečte od prvního operandu Y. Výsledkem je zbytek po dělení.

Instrukce **mod2** je podobná instrukci **mod** (viz [5E Modulo trunc, mod](#)) a pro kladná čísla poskytuje stejný výsledek. Odlišnost se projeví u záporných čísel. Operace **mod2** použije k zaokrouhlení výsledku funkci floor, tedy zaokrouhlení směrem dolů. Výsledek má stejné znaménko jako druhý operand (na rozdíl od funkce **mod**, která zachovává znaménko prvního operandu).

Instrukci **mod2** lze využít např. k normalizaci úhlu do rozsahu 0 až 359°, protože správně ošetří i záporné úhly.

### **Příklad:**

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

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

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

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

## 50 Absolutní hodnota, |x|

Symbol **|x|**

Vyvolání tlačítka **2nd |x|**

Funkce **|x|** upraví číslo na absolutní hodnotu (odstraní negativní znaménko čísla).

Je-li před stiskem **|x|** uveden prefix **INV**, provede se s obsahem registru X (číslo na displeji) znaménková operace 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 **INV |x|** lze stejně tak použít operaci **Op 10**.

## 51 Krok programu zpět, BST

Symbol **BST**

Vyvolání tlačítkem **BST**


Tlačítkem **BST** (Back Step) se v režimu programování sníží ukazatel adresy programu o 1.

Bližší informace naleznete v kapitole [Programování](#).




## 52 Režim exponentu, EE

Symbol 









Vyvolání tlačítkem 

Tlačítkem  se zapne režim exponentu. Je-li tlačítko stisknuto během zadávání čísla, přejde se na zadávání exponentu. Současně se zapne režim zobrazení ve vědecké notaci s exponentem.

Je-li tlačítko stisknuto mimo editaci čísla, zapne se režim zobrazení ve vědecké notaci s exponentem a zahájí se editace exponentu čísla. Tato funkce je často používána k odstranění skrytých číslic čísla, protože zahájením editace se do editoru načtou pouze zobrazené číslice, ne plná přesná hodnota čísla.


Stiskem prefixu  před stiskem  se ukončí režim zobrazení exponentu. Jinou možností ukončení režimu zobrazení exponentu je stisk klávesy .

### **Příklad:**

        [3.1416] ... zaokrouhlí číslo na 4 místa

## 53 Levá závorka, (


Symbol 

Vyvolání tlačítkem 

Tlačítko  zahájí výpočet části výrazu.

## 54 Pravá závorka, )


Symbol 



Vyvolání tlačítkem 

Tlačítko  ukončí výpočet části výrazu.

## 55 Dělení, :

Symbol 


Vyvolání tlačítkem 

Tlačítko  vydělí první operand druhým operandem. Jedná-li se o nejnižší úroveň výrazu, může být výpočet opakován pro jiný první operand opakovaným stiskem .

## 56 Zrušení bajtu z programu, Del

Symbol 

Vyvolání tlačítky  


Tlačítko  (Delete), stisknuté v režimu programování, smaže z aktuální pozice programu bajt. Následující data se přisunou, až od konce paměti nebo od následujícího prázdného bajtu FF. Absolutní adresy se při přesunu nepřepočítávají.



Bližší informace naleznete v kapitole [Programování](#).

## 57 Technický mód, Eng

Symbol 

Vyvolání tlačítky  

Tlačítkem  (Engineer) se aktivuje technický (inženýrský) mód zobrazení čísla. Číslo se zobrazí s exponentem, který je násobkem 3 (dekadický a oktalový mód zobrazení) nebo 4 (hexadecimální a binární mód zobrazení).

Technický mód  se nedeaktivuje stiskem . K jeho vypnutí je nutné

použít posloupnost **INV** **Eng**.

### **Příklad:**

**5** **:** ... mějme napětí 5 V

**2** **EE** **3** **+/-** ... vydělíme napětí proudem 2 mA

**=** **Eng** [2.5+3] ... hodnota rezistoru bude 2.5 kOhm

## 58 Zaokrouhlení, Fix

Symbol **Fix**

Vyvolání tlačítka **2nd** **(**

Pomocí tlačítka **Fix** se zaokrouhlí číslo zobrazené na displeji na zadany počet desetinných míst. Jako parametr se zadá číslice, představující počet desetinných míst za desetinnou tečkou. Číslice může být **0** až **9** (představující 0 až 9 desetinných míst), ale také hexadecimální číslice **0A** až **0D**, která znamená 10 až 13 desetinných míst.

V režimu zaokrouhlování se číslo doplní zprava nulami, až do zadaného počtu desetinných míst. Zadáním posloupnosti **INV** **Fix** nebo **Fix** **0F** se zaokrouhlování vypne. V tom případě se číslo zobrazí v plné přesnosti a koncové nevýznamné nuly se odstraní.

U originálního kalkulátoru TI-58/59 se k vypnutí zaokrouhlování používala posloupnost **Fix** **9**. Ta je u ET-58 považována za platné nastavení 9 desetinných míst. Při importu programu z TI-58/59 je nutné provést korekci a nahradit **Fix** **9** kódem **INV** **Fix** nebo **Fix** **0F**.

Zaokrouhlení ovlivní pouze zobrazení čísla. Vnitřně se s číslem i nadále počítá v plné přesnosti. Je-li potřeba skryté číslice skutečně odstranit, lze to provést pomocí tlačítka **EE** (viz [52 Režim exponentu, EE](#)).

Nastavený režim zaokrouhlování ovlivňuje i způsob zobrazení velmi malých čísel. Je-li zapnuto zaokrouhlování, a není-li zapnut režim exponentu, zobrazí se na displeji nuly pro malá čísla, a to i v případě, že platné číslice přetekly za pravou hranici displeje. Není-li zapnuto zaokrouhlování,

kalkulátor přejde na zobrazení s exponentem v případě, že exponent je menší než -3.

## 59 Celé číslo, Int

Symbol **Int**

Vyvolání tlačítka **2nd** **)**

Tlačítkem **Int** (Integer) lze z čísla odstranit číslice za desetinnou tečkou neboli oříznutí čísla na celé číslo. Funkce má stejný význam jako zaokrouhlení směrem k nule.

Je-li před povelom **Int** použit prefix **INV**, provede se inverzní funkce - odstranění celé části čísla a ponechání desetinné části (frac, fraction).

### **Příklad:**

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

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

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

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

## 5A Nastavení kontrastu displeje, LCD

Symbol **LCD**

Vyvolání tlačítka **2nd** **2nd** **BST**

Tlačítkem **LCD** lze upravit kontrast displeje. Po stisku tlačítka si instrukce vyžádá zadání číslice 0 až 9. Číslo 0 představuje minimální kontrast (písmo je šedé nebo sotva viditelné), číslo 9 je maximální kontrast (pod znaky jsou tmavé obdélníky).

Kontrast LCD displeje je závislý na napájecím napětí. Nastavením kontrastu je možné se dostat až mimo viditelnou oblast. V tom případě



může být nutné nastavit kontrast displeje po paměti. Nejsou-li na displeji viditelné žádné znaky (kalkulátor vypadá jako vypnutý), zkuste z klávesnice nastavit vyšší kontrast displeje použitím vyššího čísla. Případně stiskněte nejdříve **CLR**, možná je kalkulátor ve vypnutém stavu. Naopak jsou-li na displeji černé obdélníky, nastavte nižší hodnotu kontrastu displeje.

Uvedením prefixu **INV** před instrukcí **LCD** lze zjistit aktuálně nastavenou hodnotu kontrastu displeje (v registru X je navracena hodnota 0 až 9).

## 5B Posun doleva, SHL

Symbol **SHL**

Vyvolání tlačítka **2nd** **2nd** **EE**

Tlačítkem **SHL** lze první argument Y (v zásobníku operací) posunout doleva o počet bitů daný druhým argumentem X (číslo na displeji). Operace posunu o 1 bit doleva odpovídá operaci násobení číslem 2.

### **Příklad:**

**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 Posun doprava, SHR

Symbol **SHR**

Vyvolání tlačítka **2nd** **2nd** **(**

Tlačítkem **SHR** lze první argument Y (v zásobníku operací) posunout doprava o počet bitů daný druhým argumentem X (číslo na displeji). Operace posunu o 1 bit doprava odpovídá operaci dělení číslem 2.

### **Příklad:**

**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 Zaokrouhlení, round

Symbol **round**

Vyvolání tlačítka **2nd** **2nd** **)**

Tlačítkem **round** se provede zaokrouhlení čísla na displeji na nejbližší celé číslo. Je-li desetinná část vyšší nebo rovna 0.5, číslo se zaokrouhlí nahoru. Je-li desetinná část menší než 0.5, zaokrouhlí se dolů.

Na rozdíl od funkce **Fix** se zaokrouhlení provádí se skutečným číslem, ne pouze pro zobrazení.

Uvedením prefixu **INV** před funkcí **round** se odstraní celá část čísla s využitím zaokrouhlení čísla směrem dolů (floor). Následkem této operace bude číslo, které bude vždy kladné a bude v rozsahu 0 (včetně) až 1 (vyjma). Pro kladná čísla je výsledek shodný s funkcí **INV** **Int** (viz [59 Celé číslo, Int](#)). Pro záporná čísla se k nenulovému výsledku přičte 1.

### **Příklad:**

**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**

Vyvolání tlačítka **2nd** **2nd** **:**

Operace modulo vydělí první operand Y (v zásobníku) druhým operandem X (na displeji), převede výsledek na celé číslo, vynásobí jím druhý operand X a odečte od prvního operandu Y. Výsledkem je zbytek po dělení.

Instrukce **mod** je podobná instrukci **mod2** (viz [4E Modulo floor, mod2](#)) a pro kladná čísla poskytuje stejný výsledek. Odlišnost se projeví u záporných čísel. Operace **mod** použije k zaokrouhlení výsledku funkci trunc, tedy zaokrouhlení směrem k nule. Výsledek má stejné znaménko jako první operand (na rozdíl od funkce **mod2**, která zachovává znaménko druhého operandu).

### **Příklad:**

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

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

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

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

## 60 Stupně, Deg

Symbol **Deg**

Vyvolání tlačítka **2nd** **x**

Tlačítko **Deg** přepne výpočty goniometrických funkcí na stupně (plný úhel je 360°).

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

## 61 Skok, GTO

Symbol **GTO**

Vyvolání tlačítkem **GTO**

Tlačítko **GTO** (Go To) slouží k provedení nepodmíněného skoku v programu. Jako parametr se zadává buď 3-místný číselný kód absolutní cílové adresy, nebo tlačítko představující návěští v programu.

Použitím klávesy **Ind** za kódem **GTO**, ale před uvedením adresy skoku, se instrukce změní na instrukci s nepřímým adresováním, **GTO Ind** (kód instrukce 83, viz [83 Nepřímý skok, GTO Ind](#)).

Je-li instrukce **GTO** použita v prováděcím režimu, nastaví se ukazatel programu na zvolenou adresu. To lze využít k přesunu ukazatele během programování (bližší viz [Programování](#)).

Bližší o způsobech adresování v kapitole [Nepřímé adresování](#).

Tlačítko **GTO** má ještě jednu speciální funkci. Podržíte-li tlačítko **GTO** během chodu programu, bude na displeji problikávat v dolním řádku aktuální obsah displeje (obsah registru X) a na horním řádku aktuální adresa programu s aktuálně prováděným příkazem. Tímto monitorováním se ovšem chod programu mnohanásobně zpomalí.

## 62 Nepřímý výběr programu knihovny, Pgm Ind

Symbol **Pgm Ind**

Vyvolání tlačítky **2nd LRN 2nd y^x**

Instrukce **Pgm Ind** umožňuje vybrat program knihovny stejně jako instrukce **Pgm** (viz [36 Výběr programu knihovny, Pgm](#)), jen namísto z parametru instrukce se číslo knihovny převezme z datového registru.

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

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

### **Příklad:**

**2 STO 01** ... do registru R01 se uloží číslo knihovny 02

**Pgm Ind 01** [ML-02 (875)] ... aktivuje se knihovní program 02

## 63 Nepřímá záměna čísla s datovým registrem, Exc Ind

Symbol **Exc Ind**

Vyvolání tlačítka **2nd RCL 2nd y^x**

Instrukcí **Exc Ind** lze zaměnit číslo na displeji s obsahem datového registru stejně jako instrukce **Exc** (viz [48 Záměna čísla s datovým registrem, Exc](#)), jen namísto z parametru instrukce se číslo registru převezme z datového registru.

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

## 64 Nepřímé násobení a dělení registru, Prd Ind

Symbol **Prd Ind**


Vyvolání tlačítka **2nd SUM 2nd y^x**



Instrukcí **Prd Ind** lze vynásobit či vydělit obsah datového registru číslem na displeji stejně jako instrukce **Prd** (viz [49 Vynásobení a vydělení datového registru, Prd](#)), jen namísto z parametru instrukce se číslo registru převezme z datového registru.

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

## 65 Násobení, x

Symbol 


Vyvolání tlačítkem 


Tlačítko  vynásobí první operand druhým operandem. Jedná-li se o nejnižší úroveň výrazu, může být výpočet opakován pro jiný první operand opakovaným stiskem .





## 66 Prodleva, Pause

Symbol 

Vyvolání tlačítky  

Příkaz , uvedený v programu, pozastaví na dobu 0.25 sekundy provádění programu a zobrazí aktuální obsah displeje (obsah registru X).



Viz též monitorování běhu programu tlačítkem  ([61 Skok, GTO](#)).



Ke krátkému pozastavení programu bez zobrazení displeje slouží příkazy   a  .

## 67 Rovno, x=t

Symbol 

Vyvolání tlačítky  

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

**x=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=t** zadán prefix **INV**, 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 porovnávání čísla zaokrouhlovat nebo testovat v intervalu s větší tolerancí.

### **Příklad nepřímého adresování:**

**RST** **LRN** ... aktivuje programovací mód

**Lbi** **A** **x=t** **Ind** **01** **CLR** **RTN** ... při shodě skočí na adresu z R01

**1** **RTN** ... větev na adrese 7 vrací hodnotu 1

**LRN** ... návrat do prováděcího módu

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

**5** **x/t** **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**

Vyvolání tlačítka **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**

Vyvolání tlačítka **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**

Vyvolání tlačítka **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.

### **Příklad:**

**RST** **LRN** ... aktivuje programovací mód

**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**

Vyvolání tlačítky **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**

Vyvolání tlačítky **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**

Vyvolání tlačítka **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 nepřímo. 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). Nejprve 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**

Vyvolání tlačítka **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'.

### **Příklad:**

**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**

Vyvolání tlačítka **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**

Vyvolání tlačítkem **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**

Vyvolání tlačítka **STO** **2nd** **y<sup>x</sup>**

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

## 73 Nepřímé vyvolání obsahu registru, RCL Ind

Symbol **RCL** **Ind**

Vyvolání tlačítka **RCL** **2nd** **y<sup>x</sup>**

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

## 74 Nepřímé přičtení a odečtení čísla z registru, SUM Ind

Symbol **SUM** **Ind**

Vyvolání tlačítka **SUM** **2nd** **y<sup>x</sup>**


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 







Vyvolání tlačítkem 




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 

Vyvolání tlačítka  

Instrukcí  lze označit místo v programu jako návěští. Za kódem  následuje jako parametr kód tlačítka, použitého jako návěští. Lze použít jakékoliv tlačítko kromě , číslic  až  a kromě .

Na místo programu označené návěštím lze skákat pomocí skokových instrukcí s uvedenou adresou, jako je , ,  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 relokace 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 . Jako parametr  se zadají 2 HEX číslice. Návěští může mít kód 10 až 0FE (vyjma 40 = ). Obzvláště jsou k tomu vhodné kódy 0A0 až 0FE, které nejsou kalkulátorem využity a mají význam prázdných operací. Ovšem instrukci  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  **$x \geq t$**

Vyvolání tlačítka **2nd** **4**

Instrukce  **$x \geq t$**  umožňuje porovnat registr X (obsah displeje) s pomocným registrem T (nastaveným tlačítkem  **$x < t$** ). 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  **$x \geq t$** .

## 78 Statistika, Stat

Symbol **Stat**

Vyvolání tlačítka **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  $x*y$

## 79 Průměr, Mean

Symbol **Mean**

Vyvolání tlačítka **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** =  $\sqrt{(\text{sum}(x^2) - \text{sum}(x)^2/N)/(N-1))}$ .

### **Příklad:**

**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**

Vyvolání tlačítka **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

### ***Příklad - 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 =

... Poznámka: **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**

Vyvolání tlačítky **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 opakováním stiskem **=**.

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

### **Příklad:**

**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

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

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

	10110011
XOR	00100110
=	10010101

## 80 Grady, Grad

Symbol **Grad**

Vyvolání tlačítka **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**

Vyvolání tlačítkem **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**

Vyvolání tlačítka **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  $\leq 0$
- 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**.*

### **Příklad - 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**

Vyvolání tlačítky **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](#).

### Příklad

**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**

Vyvolání tlačítky **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 **+**

Vyvolání tlačítkem **+**

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**

Vyvolání tlačítka **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.

**Poznámka:** 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**

Vyvolání tlačítka **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.

### ***Příklad - nepřímé zobrazení stavu přepínače:***

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

**Lbl** **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**

Vyvolání tlačítka **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.

### **Příklad, součet času:**

**1** **2** **.** **3** **0** **2** **3** ... čas je 12:30:23 (12 hodin, 30 minut a 23 sekund)

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


+ **3** **.** **4** **5** **1** **2** **DMS** [3.7533...] ... plus 3 hodiny, 45 minut a 12 sekund

  [16.1535] ... výsledný čas 16 hodin, 15 minut a 35 sekund

## 89 Ludolfovo číslo, pi

Symbol 


Vyvolání tlačítka  


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



## 8A Operace s registry, Reg







Symbol 

Vyvolání tlačítka   

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

Za instrukcí  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  před instrukcí  se provede inverzní nebo alternativní operace.

Uvedením kódu  za prvním parametrem instrukce , ale ještě před zadáním druhého parametru, se instrukce změní na nepřímou instrukci   (číselný kód 6C, viz kapitola [6C Nepřímá operace s registry, Reg Ind](#)). Nepřímá instrukce   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

### **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.

*Poznámka: Instrukce **Reg** umožňuje zadat parametr s hodnotou 0FF (operace 0FF = načtení konstanty do HIR registru H15). Pokud možno, vyhněte se používání bajtu 0FF. Při přesunech paměti s **Ins** nebo **Del** by byl parametr 0FF považován za prázdné místo a kód by byl poškozen.*

### **Příklad:**

**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**

Vyvolání tlačítka **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**

Vyvolání tlačítka **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**

Vyvolání tlačítka **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**

Vyvolání tlačítka **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 opakováním stiskem **=**.

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



### **Příklad:**

**BIN** ... přepnutí na bitový mód zobrazení

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

**OR** ... provede se bitový součet

**00100110** ... 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**

Vyvolání tlačítkem **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**

Vyvolání tlačítky **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 **.**

Vyvolání tlačítkem **.**

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 **+/-**


Vyvolání tlačítkem **+/-**



Tlačítko **+/-** změní znaménko čísla na displeji. Je-li stisknuto během zadávání exponentu čísla, změní znaménko exponentu.


## 95 Provedení výpočtu, =

Symbol **=**





Vyvolání tlačítkem **=**



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.*



### **Příklad:**

    [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 

Vyvolání tlačítka  

Instrukce  (Decrement and Skip if Zero) slouží k opakovanému provádění programu pomocí smyčky s čítáním průchodů v registru. Za instrukcí  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  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 čítecího 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.

### **Příklad - 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

**Lbi 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, opakuji 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

*Poznámka: 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**

Vyvolání tlačítka **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**

Vyvolání tlačítka **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**

Vyvolání tlačítka **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.

*Poznámka: 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**

Vyvolání tlačítka **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.

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.

### **Příklad:**

**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 **%**

Vyvolání tlačítka **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 opakovaným 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 rovnítka [=].

### 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][%] [67.65] ...  $123 - 45\% = 123 - 123 \cdot 45/100 = 67.65$

3) Spolu s operátorem násobení [\*] vypočte počet procent ze základu.

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

4) Spolu s operátorem dělení [/] vypočte, kolik procent je ze základu.

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

5) Procento [%] na pozici operátoru vypočte procenta ze základu.

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



## 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](#).

#### ***Příklad "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.

*Poznámka: Dodatečnou změnou obsahu registru X se obsah displeje nezmění.*

### **Příklad, 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.

*Poznámka: 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.

*Poznámka: 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.

*Poznámka: 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 |x|**.

## 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  $\text{var}Y = \text{sum}(y^2)/N - (\text{sum}(y)/N)^2$ .

### Příklad:

**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...

### Příklad 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  $b = (\text{sum}(y) - m \cdot \text{sum}(x))/N$ .

### Příklad:

**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)

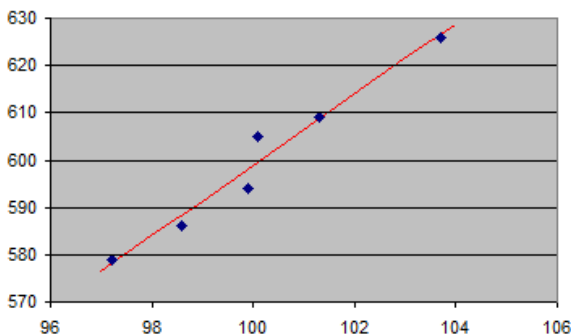
**9 9 . 9 x<>t 5 9 4 Stat** ... bod 4 (99.9, 594)

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

### **Příklad:**

... 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](#).

### **Příklad:**

... 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 regrese přímky, podle vzorce  $x = (y - b)/m$ . Viz [Op 12 Koeficienty lineární regrese](#).

### **Příklad:**

... 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**.

**Příklad** - 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**.

*Poznámka: 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**.

*Poznámka: 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.

**Příklad** - 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**.

### **Příklad - 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 navracen 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.

### ***Příklad - 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 znaku](#)). 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**).

### **Příklad:**

**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  knihovny ML-01.

**Standardní výchozí font '0', znaky 00 až 99**



**Font pro levý sloupec '1', předdefinované znaky 92 až 99**



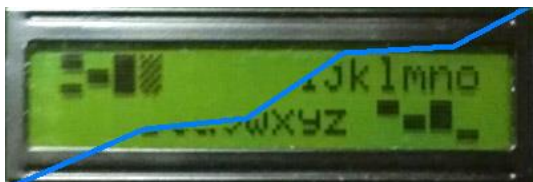
Font pro pravý sloupec '2', předdefinované znaky 92 až 99



Font pro linky a grafy '3', předdefinované znaky 92 až 99



Font pro pixely '4', předdefinované znaky 92 až 99



### **Příklad - 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

**Lbi** **=** ... 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

**Lbi** **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  
**Lbi 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.

### ***Příklad - přibývající levý ukazatel na 1. řádku:***

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

**CLR** ... počáteční hodnota ukazatele = 0

**Lbi 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.

### ***Příklad - 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.

### ***Příklad - 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. řádku za běhu

Operace **Op 48** načte do LCD displeje font číslo 2 (pravý sloupec) a zobrazí na 1. řádku za běhu programu pravý ukazatel (progress bar). Vstupem je hodnota X (obsah displeje) = 0 až 80.

### ***Příklad - 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.

### ***Příklad - 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.

### ***Příklad - 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

**Lbl 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.

### ***Příklad - 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 sloupcečku

**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.

### **Příklad - 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 sloupceku

**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.

### ***Příklad - náhodné pixely:***

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

**6 INV rand Int X/T** ... náhodná vertikální souřadnice 0...5

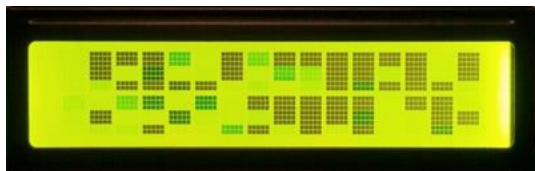
**1 6 INV rand Int** ... náhodná horizontální souřadnice 0...15

**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.

### **Příklad, 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] ... číselník 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.

*Poznámka: Registr generátoru náhody je při každém resetu kalkulatoru ukládán do EEPROM paměti a tím je zajištěna náhodnost generovaných hodnot téměř bez opakování i při opakovaných startech kalkulatoru. 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. Příklad 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	

*Poznámka: 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 dekódovat desetinné číslo. Editor čísla zůstane v editovatelném stavu, a tak lze přidávat další znaky.

*Poznámka: 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.*

### Příklad výzvy "Enter a[i].:"

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

**Lb** **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 zadán 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 čítel '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

Operaci **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

Operaci **Op 57** je na vrchol zásobníku komplexních čísel a zlomků přidáno nové číslo. Na vstupu je v registru T reálná část komplexního čísla nebo čítel zlomku, v registru X (displej) je imaginární část komplexního čísla nebo jmenovatel zlomku.

## Op 58 Načtení čísla ze zásobníku komplexních čísel

Operaci **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 čítel zlomku, v registru X (displej) je imaginární část komplexního čísla nebo jmenovatel zlomku.

## Op 59 Zrušení čísla ze zásobníku komplexních čísel

Operaci **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  $\text{asin}(X)$ . Výsledek ponechá na pozici původního čísla  $X$ .

### Op 6C Arkus kosinus komplexního čísla $\text{acos}(X)$

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 návěštím **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 návěšním **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 čísel 'a' a v registru T navrátí jmenovatel 'b' zlomku. 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.

## 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  $X \cdot Y$  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 kalkulátoru při nečinnosti, v sekundách. 0 znamená vyžazené automatické vypínání.

## Op 8A Zobrazení verze firmware kalkulátoru

Operací **Op 8A** zobrazí na displeji verzi firmware kalkulátoru stejně jako po resetu. Na displeji se na 1 sekundu 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.

## Op 8B Reset kalkulátoru

Operací **Op 8B** resetuje kalkulátor stejně, jako vyjmutí baterie.

## 16. Tabulka znaků

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. Znaký 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 |.

### ***Příklad, 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

<b>Build</b>	<b>Key-Dec</b>	<b>Flags</b>	<b>Fill</b>	<b>Reset</b>
<b>Font</b>	<b>Key-Hex</b>	<b>Stat</b>	<b>Clear</b>	<b>Diag</b>

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).

### **Příklad:**

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  $A * x$  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.

### **Příklad**

Součet matic  $C = A - 2 \cdot B$

<b>A =</b>	<b>2</b>	<b>3</b>	<b>0</b>
	<b>1</b>	<b>0</b>	<b>5</b>
<b>B =</b>	<b>4</b>	<b>0</b>	<b>-1</b>
	<b>3</b>	<b>2</b>	<b>6</b>

**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 =      -10      6**  
**-43      -34**

## 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.

**Příklad, výpočet  $((2 + 3i) \cdot (1 - i))^{(1 + i)}$ :**

**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  $X \cdot Y$

**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.

### ***Příklad, 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  $\sqrt{x}$

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.

### **Příklad, výpočet $\sin(1 + 3i)$ :**

**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'.

**Příklad, 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 'x2', do které maximálně má probíhat vyhledání nuly, a stiskněte **[x<>]**. Zadejte počáteční hodnotu 'x1', 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 'x1'. To proto, aby se opakovaně nenacházela naposledy nalezená hodnota x. Leží-li průchod nulou v počáteční hodnotě 'x1', zvolte nižší výchozí 'x1'.

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.

**Příklad, průchody nulou funkce  $f(x) = 4 \cdot \sin(x) + 1 - x$ :**

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

**Lbl A'** ... označení návěští začátku funkce **A'**

**( STO 10** ... úschova hodnoty 'x' do registru R10

**sin x 4** ...  $4 \cdot \sin(x)$

**+ 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.

**Příklad, 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  $\pi$



**2** **\*** **pi** **=** **B** **+/-** **A** **E** ... vykreslení grafu v intervalu  $-2\pi$  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.

### **Příklad:**

**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

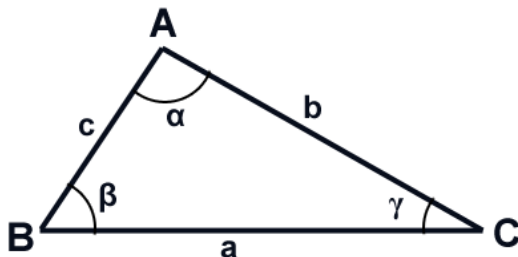
**1 R/S 8 R/S 2 7 R/S 6 4 R/S 1 2 5 R/S** ... zadání 5 hodnot

**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.

### **Příklad:**

**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...

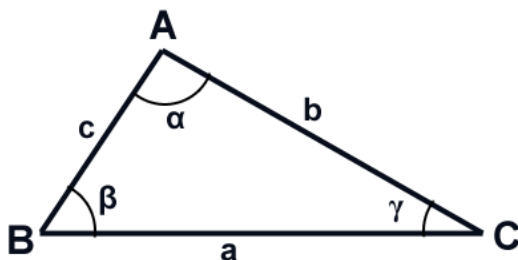
**R/S** [84.978...] ... vypočtený úhel A = 84.978...

**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.

### **Lb| 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**.

### **Lb| 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.

### **Příklad:**

**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...

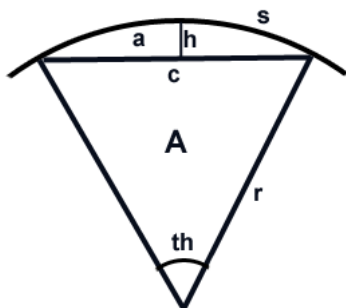
**R/S** [16.7523...] ... vypočtený obvod 16.7523...



## 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  $\theta$ )

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 $\theta$ a délka tětiny $c$

Po stisku **C** zadejte středový úhel ' $\theta$ ' 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 ' $\theta$ ', 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 ' $\theta$ ', délku oblouku ' $s$ ', obsah výseče ' $A$ ', obsah úseče ' $a$ ' a výšku úseče ' $h$ '. Pokračování stiskem **R/S**.

### **Příklad:**

**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  $h = 2.1424...$

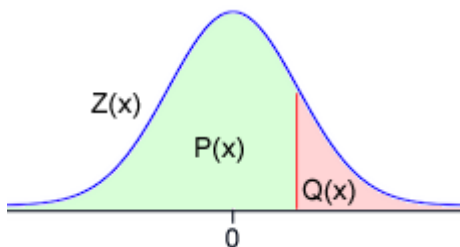
## ML-14 Normální rozdělení

**GRAPH**  
**Z(x)**

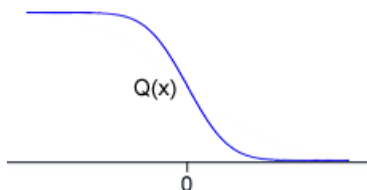
**GRAPH**  
**Q(x)**

**GRAPH**  
**P(x)**

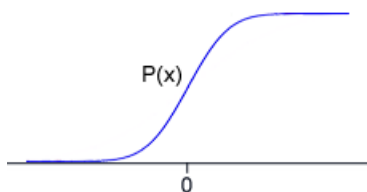
Program ML-14 počítá standardní normální (neboli Gaussovo) rozdělení pravděpodobnosti.



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).

### **Lb| 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'.

### **Lb| 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'.

### **Lb| C Rovnoměrná náhodnost**

Funkce **C** generuje náhodné desetinné číslo s rovnoměrným rozložením v intervalu od 'A' (včetně) do 'B' (vyjma).

### **Lb| D Rovnoměrná celočíselná náhodnost**

Funkce **D** generuje náhodné celé číslo s rovnoměrným rozložením v intervalu od 'A' (včetně) do 'B' (vyjma).

### **Lb| 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
x!	ln x!	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  $H(n) = n^{n-1} * (n-1)^{(n-1)} * \dots$ . 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**.

*Poznámka: Před následujícími příklady proveďte 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.*



### **Příklad 1:**

V policiče jsou 4 knihy. Kolik je možností uspořádání knih?

☐ ☐ ☒ A ... zadání celkového počtu prvků  $n = 4$

☐ D [24] ... počet permutací bez opakování = 24

### **Příklad 2:**

25 studentů se uchází o stipendium. Pouze 3 nejlepší mohou být vybráni a podle pořadí dostanou různou výšku stipendia. Kolik je možných výsledků?

☐ ☐ ☐ ☒ A ... zadání celkového počtu prvků  $n = 25$

☐ ☐ ☒ B ... zadání počtu prvků výběru  $r = 3$

☐ C [13800] ... počet variací bez opakování = 13800

### **Příklad 3:**

V balíčku je 52 karet. Hráč si vybere 4 karty. Kolik je možností výběru?

☐ ☐ ☐ ☒ A ... zadání celkového počtu prvků  $n = 52$

☐ ☐ ☒ B ... zadání počtu prvků výběru  $r = 4$

☐ E [270725] ... počet kombinací bez opakování = 270725

### **Příklad 4:**

V abecedě je 26 písmen. Kolik z nich můžeme sestavit slov o 3 písmenech?

☐ ☐ ☐ ☒ A ... zadání celkového počtu prvků  $n = 26$

☐ ☐ ☒ B ... zadání počtu prvků výběru  $r = 3$

☐ C [17576] ... počet variací s opakováním = 17576

### **Příklad 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:  $123456789! = 2.853512521_{7299} * 10^{945335859}$ .

*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. Celočíselná 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  $100\$ = 2.7031768576518 * 10^{6940}$ .

### **Příklad 8:**

Hyper-faktoriál čísla 25:

**2 5 D"**

Výsledek  $H(25) = 4.9718576220366 * 10^{386}$ .

## 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.

### **Příklad:**

**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

<b>S</b>	<b>(1+i)S</b>	<b>a</b>	<b>(1+i)a</b>	<b>INIT</b>
<b>N</b>	<b>%I</b>	<b>PV</b>	<b>FV</b>	

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.

### **Příklad 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

### **Příklad 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

### **Příklad 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

### **Příklad 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.

### **Příklad 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

### **Příklad 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%

### ***Příklad 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

### ***Příklad 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).

### **Lbl 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.

### **Příklad 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

### **Příklad 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.

### **Příklad:**

**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 desetin, jsou přidány jako další číslice za údajem vteřin.

### **Lbl A** Zadání prvního čísla

Funkcí **A** lze zadat první číslo časového nebo úhlového údaje. Číslo se zadává ve formátu dd.mmss.

### **Lbl 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**.

### **Lbl 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**.

### **Lbl 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**.

### **Příklad:**

**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.

**Příklad, 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

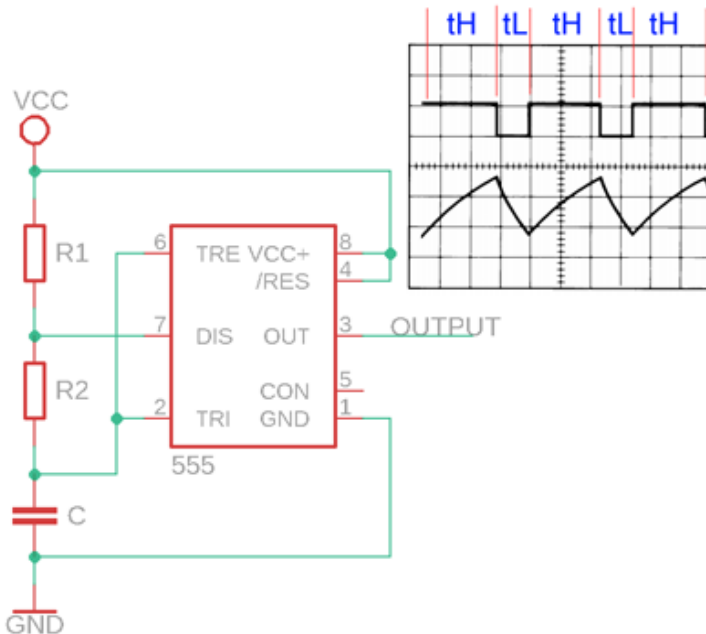
**C** ... součet čísel, na displeji výsledek 25/6



## 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.

### **Příklad:**

**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

### **Příklad:**

**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 XL v ohmech (f a L musí být známe)

**Lbl** **D'** Přepočet induktivní reaktance XL na indukčnost L (f musí být známe)

**Lbl** **E** Výpočet kapacitní reaktance XC v ohmech (f a C musí být známe)

**Lbl** **E'** Přepočet kapacitní reaktance XC na kapacitu C (f musí být známe)

### **Příklad:**

**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 XC = 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  $X_p$

### **Příklad:**

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 kapacity  $C = 25$  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.

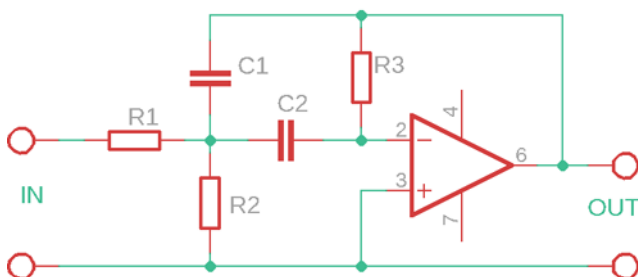


## ML-31 (EE-13) Aktivní filtr

C1	C2	->BP	->LP	->HP
alpha	A	F	B	

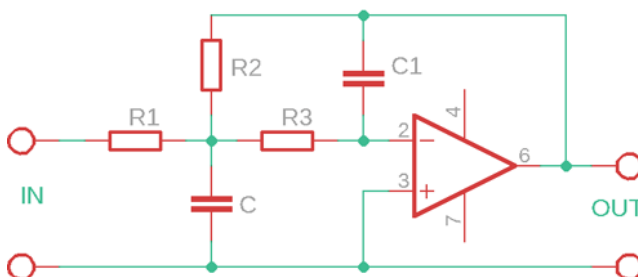
Program ML-31 slouží k výpočtu aktivních filtrů s operačním zesilovačem - pásmová propust BP, dolní propust LP a horní propust HP.

### Aktivní pásmová propust (Bandpass BP)



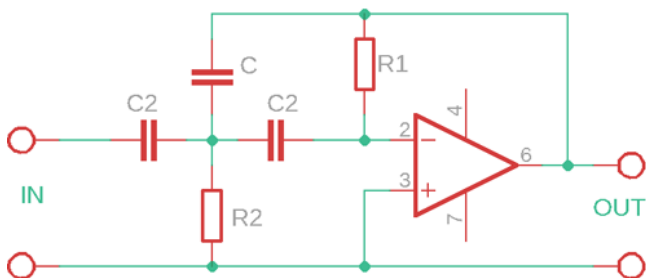
**Znamé údaje:** Šířka 3-dB pásma 'B' v Hz, zisk ve středu pásma 'A' v dB, frekvence středu pásma 'F' v Hz, kapacity C1 a C2 ve faradech. Program spočítá hodnoty R1, R2 a R3.

### 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**.

#### **Příklad 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

#### **Příklad 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  $R_3 = 511.4... \text{ ohm}$

### **Příklad 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  $C = 23.55... \text{ nF}$

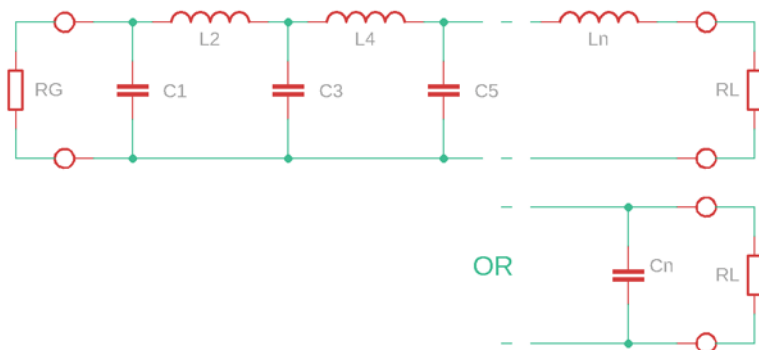
**R/S** [84.496...+3] ... výpočet  $R_1 = 84.496... \text{ kohm}$

**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  $\epsilon_{ps}$  v dB

Zvlnění  $\epsilon_{ps}$  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**.

### **Příklad 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  $C9 = 2.763... \text{ nF}$

### **Příklad 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**.

### **Příklad:**

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  $y(0.2) = 0.2960...$

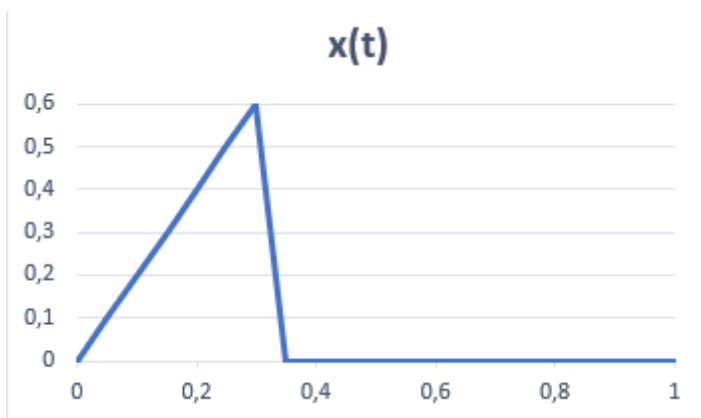
**R/S** [0.5808...] ... výpočet  $y(0.3) = 0.5808...$

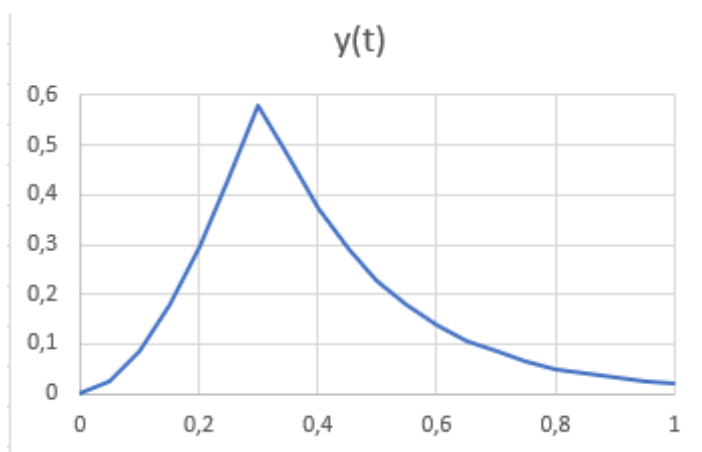
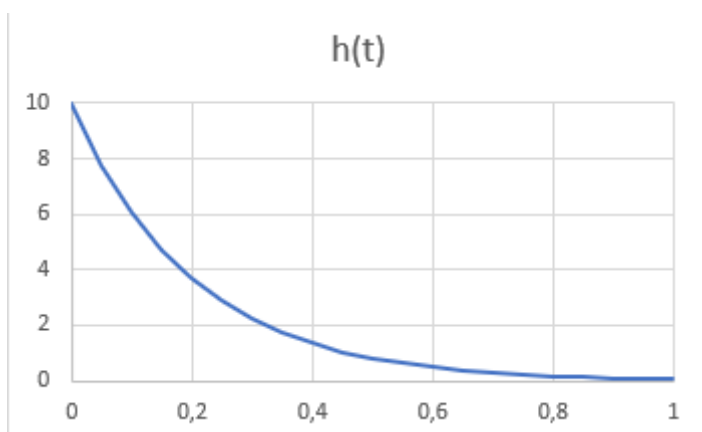
**R/S** [0.3978...] ... výpočet  $y(0.4) = 0.3978...$

**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**.

### **Příklad:**

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. Doplňkově 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

### Příklad

**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  $1 \text{ W}$  by byl odpor  $R = 25 \text{ ohmů}$

**C** ... vypočtený odpor  $R = 25 \text{ ohmů}$  se použije na vstupu

**B'** [200-3] ... odporem  $25 \text{ ohmů}$  by při  $5 \text{ V}$  tekla proud  $200 \text{ mA}$

## ML-36 Sérové a paralelní řazení

RLC +RLs,Cp +RLp,Cs

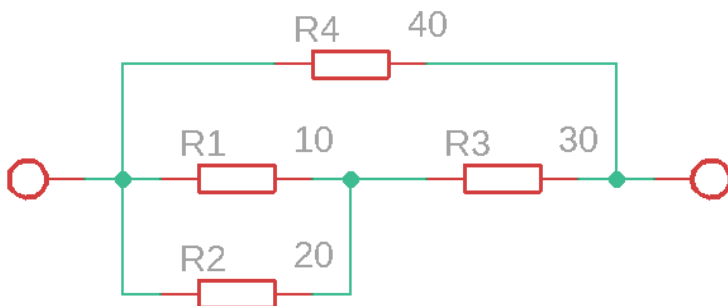
Program ML-36 počítá výslednou hodnotu odporu, indukčnosti nebo kapacity, spojené sériově či paralelně.

**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

**Příklad:**



**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).

### **LbI 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.

*Poznámka: 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.*

### **LbI B** Zadání výchozího data ve formátu MM.DD

### **LbI C** Zadání výchozího času ve formátu HH.MMSS

### **LbI D** Zadání přírůstku času 'dt' ve formátu HH.MMSS

Maximální povolená hodnota 'dt' je +- 648 hodin (tj. 27 dnů).

### **LbI E** Výpočet cílového času ve formátu HH.MMSS

### **LbI 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.

### **Příklad 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

## **Příklad 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

### **Příklad 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

### **Příklad:**

**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 2 R/S** ... zadání dat d1 = 5.12

**1 1 R/S** ... zadání dat d2 = 11

**9** **|** **2** **R/S** ... zadání dat  $d_3 = 9.2$

**4** **|** **3** **+/-** **R/S** ... zadání dat  $d_4 = -4.3$

**1** **|** **4** **5** **+/-** **R/S** ... zadání dat  $d_5 = -1.45$

**|** **4** **R/S** ... zadání dat  $d_6 = 0.4$

**3** **7** **R/S** ... zadání dat  $d_7 = 37$

**|** **1** **R/S** ... zadání dat  $d_8 = 0.1$

**8** **|** **3** **R/S** ... zadání dat  $d_9 = 8.3$

**C** ... setřídění dat

**0** **D** ... zobrazení dat od indexu 0

**[-4.3]** ... zobrazení dat  $d_0 = -4.3$

**R/S** **[-1.45]** ... zobrazení dat  $d_1 = -1.45$

**R/S** **[0.1]** ... zobrazení dat  $d_2 = 0.1$

**R/S** **[0.4]** ... zobrazení dat  $d_3 = 0.4$

**R/S** **[5.12]** ... zobrazení dat  $d_4 = 5.12$

**R/S** **[8.3]** ... zobrazení dat  $d_5 = 8.3$

**R/S** **[9.2]** ... zobrazení dat  $d_6 = 9.2$

**R/S** **[10.6]** ... zobrazení dat  $d_7 = 10.6$

**R/S** **[11]** ... zobrazení dat  $d_8 = 11$

**R/S** **[37]** ... zobrazení dat  $d_9 = 37$

**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.

### **Příklad:**

**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**

### **Příklad:**

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  $n$  ( $0 \dots N-1$ )

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**.

### Příklad:

**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  $(x_4, y_4) = (1.1, 0.3643)$

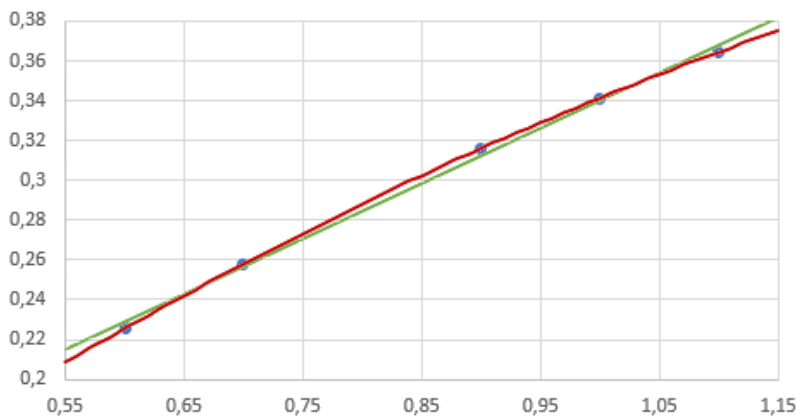
**8 C** [0.28811] ... interpolace  $f(0.8) = 0.28811$

*Poznámka: 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*

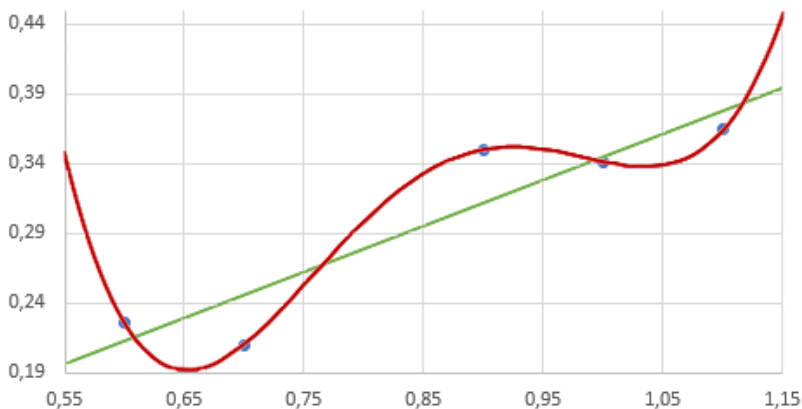


*datovými body, i za cenu zvlněného průběhu křivky.*

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.

**Příklad,  $f(x) = x^3 - x^2 - x + 2$ :**

**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  $f(x) = 2.1851...$

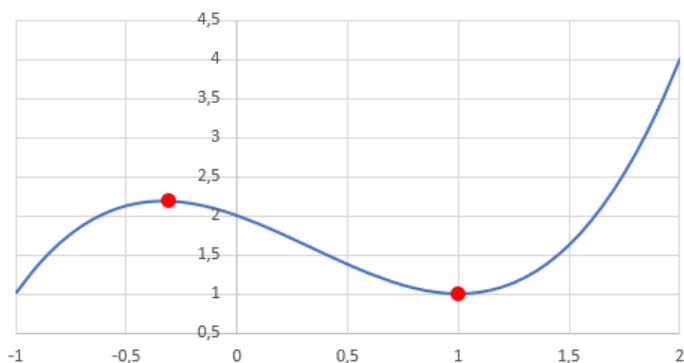
**x<>t** [-1] ... kritický bod je maximum

**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

### **Lb| 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

### **Lb| 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

### **Lb| 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

#### **Lbl 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

#### **Lbl 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

#### **Lbl 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

### **Příklad:**

**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.

#### **Lb** **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.

#### **Lb** **E** Start nové hry

*Poznámka: Pro reprodukovatelnost náhody je možné inicializovat generátor náhody operací **Op** **52**. Běžně není potřeba generátor náhody inicializovat.*

### **Příklad:**

**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**

*Poznámka: Pro reprodukovatelnost náhody je možné inicializovat generátor náhody operací **Op 52**. Běžně není potřeba generátor náhody inicializovat.*

### **Příklad:**

**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)

*Poznámka: Pro reprodukovatelnost náhody je možné inicializovat generátor náhody operací **Op 52**. Běžně není potřeba generátor náhody inicializovat.*

**Příklad:**

**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/sec<sup>2</sup>. S klesající hmotností modulu účinnost motorů roste. Gravitace na Marsu je 13 stop/sec<sup>2</sup>.

**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

**Příklad:**

**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.

*Poznámka: 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**

**Příklad:**

**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

### **Příklad:**

**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  $45^\circ$ .

... 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  $45^\circ$ .

... 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  $45^\circ$ .

... 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  $45^\circ$ .

... 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