

Проект
«Эмулятор микрокалькулятора «Электроника МК-61s mini»
https://github.com/UN7FGO/MK61S_MINI

Руководство по работе с терминалом

Подключение к терминалу производится через последовательный порт ПК при помощи любой программы терминала, например Putty, TeraTerm или терминала ArduinoIDE. Характеристики соединения:

- 115200 Bod
- 8 bit + 1 bit четности

Предпочтительный размер окна терминала, 40 строк по 80 символов в строке. Для корректного отображения символов кириллицы, установите кодовую страницу Win1251.

**Небольшое лирическое отступление, о терминале и том, что
мы в нем наделали**

Ввиду того, что само устройство, как и оригинал, не обладают большим дисплеем и не в состоянии отображать много информации одновременно, было решено создать инструмент, который позволит получить доступ к “внутреннему миру” устройства в более комфортном виде. Для этих целей и был выбран терминал, как наиболее простой и интуитивный способ общения с устройством.

Терминал дает возможность доступа к внутренней информации устройства, которая в оригинале была скрыта от пользователя или представлена в совсем уж минималистичном виде.

При работе с устройством, необходимо помнить, что его внутренняя организация, отличается от привычной нам. Программируемые калькуляторы 80-х годов прошлого века, создавались по Гарвардской архитектуре. Это означает, что память программы и память данных, это два разных устройства. В нашем случае, эта память не только разная, но она и организована по разному. Так, если память программы имеет байтовую организацию, то память данных имеет структуру вещественного числа и содержит мантиссу и порядок числа.

Для удобства работы с программным кодом, мы ввели в терминал, возможность работы с программным кодом, на языке ассемблера. Однако и тут у нас не обошлось без нюансов, связанных со структурой команд самого программируемого калькулятора. Особенность заключается в том, что в коде калькулятора переход на указанный адрес происходит в случае НЕВЫПОЛНЕНИЯ условия. В связи с чем команды языка ассемблера выглядят “зеркальными” к обозначениям на нажимаемых кнопках устройства.

Система команд микрокалькулятора, в классическом представлении, выглядит следующим образом:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	,	/- /	ВП	Cx	В↑	F Bx
1	+	-	×	÷	↔	F 10*	F e*	F lg	F ln	F arcsin	F arccos	F arctg	F sin	F cos	F tg	
2	F π	F √	F x²	F 1/x	F xʸ	F ○	K +	K -	K ×	K ÷	K ↔					
3	K 3	K 4	K 5	K 6	K 7	K 8	K 9	K ,	K /- /	K ВП	K Cx	K В↑				
4	Π 0	Π 1	Π 2	Π 3	Π 4	Π 5	Π 6	Π 7	Π 8	Π 9	Π A	Π B	Π C	Π D	Π ↑	
5	с/п	БП	В/О	ПП	К НОП	K 1	K 2	F X=0	F L2	F X≥0	F L3	F L1	F X<0	F L0	F X=0	
6	ИП 0	ИП 1	ИП 2	ИП 3	ИП 4	ИП 5	ИП 6	ИП 7	ИП 8	ИП 9	ИП A	ИП B	ИП C	ИП D	ИП ↑	
7	K X=0 0	K X=0 1	K X=0 2	K X=0 3	K X=0 4	K X=0 5	K X=0 6	K X=0 7	K X=0 8	K X=0 9	K X=0 A	K X=0 B	K X=0 C	K X=0 D	K X=0 ↑	
8	K БП 0	K БП 1	K БП 2	K БП 3	K БП 4	K БП 5	K БП 6	K БП 7	K БП 8	K БП 9	K БП A	K БП B	K БП C	K БП D	K БП ↑	
9	K X≥0 0	K X≥0 1	K X≥0 2	K X≥0 3	K X≥0 4	K X≥0 5	K X≥0 6	K X≥0 7	K X≥0 8	K X≥0 9	K X≥0 A	K X≥0 B	K X≥0 C	K X≥0 D	K X≥0 ↑	
A	K ПП 0	K ПП 1	K ПП 2	K ПП 3	K ПП 4	K ПП 5	K ПП 6	K ПП 7	K ПП 8	K ПП 9	K ПП A	K ПП B	K ПП C	K ПП D	K ПП ↑	
B	K Π 0	K Π 1	K Π 2	K Π 3	K Π 4	K Π 5	K Π 6	K Π 7	K Π 8	K Π 9	K Π A	K Π B	K Π C	K Π D	K Π ↑	
C	K X<0 0	K X<0 1	K X<0 2	K X<0 3	K X<0 4	K X<0 5	K X<0 6	K X<0 7	K X<0 8	K X<0 9	K X<0 A	K X<0 B	K X<0 C	K X<0 D	K X<0 ↑	
D	K ИП 0	K ИП 1	K ИП 2	K ИП 3	K ИП 4	K ИП 5	K ИП 6	K ИП 7	K ИП 8	K ИП 9	K ИП A	K ИП B	K ИП C	K ИП D	K ИП ↑	
E	K X=0 0	K X=0 1	K X=0 2	K X=0 3	K X=0 4	K X=0 5	K X=0 6	K X=0 7	K X=0 8	K X=0 9	K X=0 A	K X=0 B	K X=0 C	K X=0 D	K X=0 ↑	
F																

Та же система команд, на языке ассемблера, выглядит так:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	dot	neg	pow10	clr	push	preX
1	add	sub	mul	div	swap	e10	exp	lg	ln	asin	acos	atg	sin	cos	tg	?
2	pi	sqrt	sqr	rec	pow	rot	toM	?	?	?	toMS	?	?	?	?	?
3	inMS	mod	sgn	inM	int	frc	max	and	or	xor	not	rnd	?	?	?	?
4	sto0	sto1	sto2	sto3	sto4	sto5	sto6	sto7	sto8	sto9	stoA	stoB	stoC	stoD	stoE	?
5	hlt	jmp	ret	call	nop	?	?	jz	rpt2	jlz	rpt3	rpt1	jme	rpt0	jnz	?
6	ld0	ld1	ld2	ld3	ld4	ld5	ld6	ld7	ld8	ld9	ldA	ldB	ldC	ldD	ldE	?
7	jz[0]	jz[1]	jz[2]	jz[3]	jz[R4]	jz[5]	jz[6]	jz[7]	jz[8]	jz[9]	jz[A]	jz[B]	jz[C]	jz[D]	jz[E]	?
8	jmp[0]	jmp[1]	jmp[2]	jmp[3]	jmp[4]	jmp[5]	jmp[6]	jmp[7]	jmp[8]	jmp[9]	jmp[A]	jmp[B]	jmp[C]	jmp[D]	jmp[E]	?
9	jlz[0]	jlz[1]	jlz[2]	jlz[3]	jlz[4]	jlz[5]	jlz[6]	jlz[7]	jlz[8]	jlz[9]	jlz[A]	jlz[B]	jlz[C]	jlz[D]	jlz[E]	?
A	call[0]	call[1]	call[2]	call[3]	call[4]	call[5]	call[6]	call[7]	call[8]	call[9]	call[A]	call[B]	call[C]	call[D]	call[E]	?
B	sto[0]	sto[1]	sto[2]	sto[3]	sto[4]	sto[5]	sto[6]	sto[7]	sto[8]	sto[9]	sto[A]	sto[B]	sto[C]	sto[D]	sto[E]	?
C	jme[0]	jme[1]	jme[2]	jme[3]	jme[4]	jme[5]	jme[6]	jme[7]	jme[8]	jme[9]	jme[A]	jme[B]	jme[C]	jme[D]	jme[E]	?
D	ld[0]	ld[1]	ld[2]	ld[3]	ld[4]	ld[5]	ld[6]	ld[7]	ld[8]	ld[9]	ld[A]	ld[B]	ld[C]	ld[D]	ld[E]	?
E	jnz[0]	jnz[1]	jnz[2]	jnz[3]	jnz[R4]	jnz[5]	jnz[6]	jnz[7]	jnz[8]	jnz[9]	jnz[A]	jnz[B]	jnz[C]	jnz[D]	jnz[E]	?
F	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

По вертикали первая цифра кода команды, по горизонтали вторая.

Небольшое пояснение, как были выбраны наименования команд, с учетом “инверсии” работы команд калькулятора.

Команда калькулятора	Команда языка ассемблера	Примечание
БП	JMP	Jump
ПП	CALL	Call
В/О	RET	Return
СП	HLT	Halt
Π->X	STO	Store
X<-Π	LD	Load
X=0	JZ	Jump Zero
X≥0	JLZ	Jump Loss Zero
X<0	JME	Jump More or Equal

X=0	JNZ	Jump No Zero
-----	-----	--------------

Описание команд терминала МК-61s-mini

ver - версия ПО калькулятора

```
ver
sizeof Serial 280
MK61s ver. Nov 18 2024(13:16:42)
```

list - выводит содержимое программной памяти МК-61s-mini в шестнадцатеричном виде, с указанием адреса ячейки памяти.

```
list
00. 52 16. 43 32. 42 48. 10 64. 65 80. 11 96. 64
01. 6A 17. 87 33. 5B 49. 45 65. 0F 81. 5E 97. A8
02. 1D 18. 77 34. 37 50. 10 66. 01 82. 85 98. 65
03. 5C 19. 63 35. 6D 51. 31 67. 11 83. 6D 99. 0E
04. 18 20. 5E 36. 50 52. 6C 68. 11 84. 50 A0. 0E
05. 63 21. 27 37. 61 53. 11 69. 5E 85. 6B A1. 0E
06. 5E 22. 62 38. 69 54. 59 70. 78 86. 65 A2. 0E
07. 44 23. 43 39. 12 55. 64 71. 66 87. 11 A3. 0E
08. 62 24. 00 40. 4C 56. 65 72. 64 88. 5E A4. 0E
09. 0B 25. 42 41. 40 57. 0F 73. 11 89. 96
10. 43 26. 87 42. 64 58. 11 74. 59 90. 05
11. 00 27. 0B 43. 62 59. 59 75. 78 91. 64
12. 42 28. 42 44. 10 60. 63 76. 6B 92. 11
13. 87 29. 00 45. 44 61. 6D 77. 45 93. 5E
14. 42 30. 43 46. 65 62. 50 78. 65 94. 96
15. 00 31. 5D 47. 63 63. 27 79. 6E 95. 50
```

reg - выводит значение всех регистров памяти МК-61s-mini, в формате оригинального МК-61.

```
reg
R0 = 1.0000000 02
R1 = 1.0000000 01
R2 = 0.0000000 00
R3 = 0.1000000 00
R4 = 0.0000000 00
R5 = 6.0000000 -01
R6 = -0.7000000 01
R7 = 0.0000037 07
R8 = 0.0000099 07
R9 = 1.0000000 01
RA = 1.0000000 02
RB = -4.0000000 01
RC = 1.0000000 02
RD = Г.ГГ0Г 00 00
RE = -3.9000000 01
IP: 99
```

stk - (Stack) выводит текущее состояние стека МК-61-s-mini. Состояние регистров стека, отображается так, как оно выглядит в микрокалькуляторе.

```
stk
X1 = 6.0000000 -01
T  = 0.0000000 00
Z  = 0.0000000 00
Y  = 0.0000000 00
X  = 0.0000000 00
IP = 99
```

1302 - вывод содержимого регистра R K145ИК1302

```
1302
F08F00F08F08F08F08F08000F00F02F9FF9051C32F
```

asm - ассемблирование строки текста и ввод программы из этой строки (терминала) в программную память МК-61s-mini. **Обязательно за словом asm размещать один пробел!** В конце строки так же ставится завершающий пробел. Применяемые мнемоники могут быть просмотрены по запросу, командой ISA.

Опционально в команде **asm** может присутствовать адрес трансляции (**ТА**) в виде 4 разрядного числа.

```
asm 0005 sin cos div
Assembled to mk61 parsing address 5
from new TA (5)
Parsing: sin cos div
TA = 5 : 1C
Parsing: cos div
TA = 6 : 1D
Parsing: div
TA = 7 : 13
Parsing:
Unexpected token:

pub
00. 0
01. 0
02. 0
03. 0
04. 0
05. Fsin
06. Fcos
07. :
08. 0
09. 0
█
```

При отсутствии адреса в строке ассемблирования (по умолчанию) трансляция ведется с текущего значения **ТА**.

isa - выводит список мнемонических кодов доступных для ввода программы МК-61s-mini из терминала

```
isa
00 0      01 1      02 2      03 3      04 4
05 5      06 6      07 7      08 8      09 9
0A dot    0B neg    0C pow10   0D clr    0E push
0F preX   10 add    11 sub     12 mul    13 div
14 swap   15 e10    16 exp     17 lg     18 ln
19 asin   1A acos   1B atg     1C sin     1D cos
1E tg     1F ?      20 pi      21 sqrt   22 sqr
23 rec    24 pow    25 rot     26 toM    27 ?
28 ?      29 ?      2A toMS   2B ?      2C ?
2D ?      2E ?      2F ?      30 inMS   31 mod
32 sgn    33 inM    34 int     35 frc    36 max
37 and    38 or     39 xor     3A not    3B rnd
3C ?      3D ?      3E ?      3F ?      40 sto0
41 sto1   42 sto2   43 sto3   44 sto4   45 sto5
46 sto6   47 sto7   48 sto8   49 sto9   4A stoA
4B stoB   4C stoC   4D stoD   4E stoE   4F ?
50 hlt    51 jmp    52 ret     53 call   54 nop
```

lasm (List Assembler) - позволяет вывести в терминал содержимое программной памяти, с мнемонической расшифровкой содержимого ячеек, в формате - [адрес] [содержимое ячейки памяти] [мнемоника]

```
lasm
0000 52 ret      0030 43 sto3      0060 63          0090 05 5
0001 6A ldA      0031 5D rpt0 42   0061 6D ldD      0091 64 ld4
0002 1D cos      0032 42          0062 50 hlt      0092 11 sub
0003 5C jme 18   0033 5B rpt1 37   0063 27 ?      0093 5E jnz 96
0004 18          0034 37          0064 65 ld5      0094 96
0005 63 ld3      0035 6D ldD      0065 0F preX    0095 50 hlt
0006 5E jnz 44   0036 50 hlt      0066 01 1      0096 64 ld4
0007 44          0037 61 ld1      0067 11 sub      0097 A8 call[8]
0008 62 ld2      0038 69 ld9      0068 11 sub      0098 65 ld5
0009 0B neg      0039 12 mul      0069 5E jnz 78   0099 0E push
0010 43 sto3     0040 4C stoC      0070 78          00A0 0E push
0011 00 0        0041 40 sto0      0071 66 ld6      00A1 0E push
0012 42 sto2     0042 64 ld4      0072 64 ld4      00A2 0E push
0013 87 jmp[7]   0043 62 ld2      0073 11 sub      00A3 0E push
0014 42 sto2     0044 10 add      0074 59 jl 78    00A4 0E push
0015 00 0        0045 44 sto4      0075 78
0016 43 sto3     0046 65 ld5      0076 6B ldB
0017 87 jmp[7]   0047 63 ld3      0077 45 sto5
0018 77 jz[7]    0048 10 add      0078 65 ld5
0019 63 ld3      0049 45 sto5      0079 6E ldE
0020 5E jnz 27   0050 10 add      0080 11 sub
0021 27          0051 31 mod      0081 5E jnz 85
0022 62 ld2      0052 6C ldC      0082 85
0023 43 sto3     0053 11 sub      0083 6D ldD
0024 00 0        0054 59 jl 64      0084 50 hlt
0025 42 sto2     0055 64          0085 6B ldB
0026 87 jmp[7]   0056 65 ld5      0086 65 ld5
0027 0B neg      0057 0F preX      0087 11 sub
0028 42 sto2     0058 11 sub      0088 5E jnz 96
0029 00 0        0059 59 jl 63    0089 96
```

hout (Hex Output) - выдает содержимое памяти микрокалькулятора в виде строки, содержащей начальный адрес и данные в виде набора двухсимвольных шестнадцатеричных чисел.

```

hout
0000 526A1D5C18635E44620B430042874200438777635E276243
0024 0042870B4200435D425B376D506169124C40646210446563
0048 104510316C115964650F1159636D5027650F0111115E7866
0072 641159786B45656E115E856D506B65115E960564115E9650
0096 64A8650E0E0E0E0E0E9B

```

hin [адрес] [значение] (Hex Input) - помещает указанное значение по указанному адресу. Адрес указывается в четырехзначном формате. Значение указывается в двузначном-шестнадцатеричном виде.

```

hin 0000 526A1D5C18635E44620B430042874200438777635E276243
parsing address 0

52,6A,1D,5C,18,63,5E,44,62,B,43,0,42,87,42,0,43,87,77,63,5E,27,62,43,
Stream received!

```

Данные для команды **hin**, могут быть получены командой **hout**, что позволит вам поделиться своей программой с другим человеком.

ins (Insert) – вставляет в программу, по указанному адресу операцию с указанным кодом, при этом сдвигая все последующие команды и корректирует условные и безусловные переходы.

Ins [адрес] [код команды]

Адрес указывается в десятично-шестнадцатеричном виде, код команды в шестнадцатеричном.

```

lasm
0000 34 int
0001 40 sto0
0002 01 1
0003 60 ld0
0004 12 mul
0005 5D rpt0 03
0006 03
0007 50 hlt
0008 00 0

ins 0004 50
lasm
0000 34 int
0001 40 sto0
0002 01 1
0003 60 ld0
0004 50 hlt
0005 12 mul
0006 5D rpt0 03
0007 03
0008 50 hlt
0009 00 0

```

pub (Publication) - выводит в терминал содержимое памяти микрокалькулятора, в формате, подобном формату публикации программ в журналах.

```
pub
00. В/О      30. X->ПЗ      60. 63      90. 5
01. П->ХА    31. FL0       61. П->ХД    91. П->Х4
02. Fcos     32. 42      62. С/П     92. -
03. Fx<0    33. FL1       63. ?       93. Fx=0
04. 18       34. 37      64. П->Х5    94. 96
05. П->Х3    35. П->ХД    65. Вх      95. С/П
06. Fx=0     36. С/П     66. 1       96. П->Х4
07. 44       37. П->Х1    67. -       97. РъПП8
08. П->Х2    38. П->Х9    68. -       98. П->Х5
09. /-/      39. *       69. Fx=0    99. В^
10. X->ПЗ    40. X->ПС    70. 78      A0. В^
11. 0        41. X->П0    71. П->Х6    A1. В^
12. X->П2    42. П->Х4    72. П->Х4    A2. В^
13. КБП7     43. П->Х2    73. -       A3. В^
14. X->П2    44. +       74. Fx>=0   A4. В^
15. 0        45. X->П4    75. 78
16. X->ПЗ    46. П->Х5    76. П->ХВ
17. КБП7     47. П->Х3    77. X->П5
18. Kx≠0 7   48. +       78. П->Х5
19. П->Х3    49. X->П5    79. П->ХЕ
20. Fx=0     50. +       80. -
21. 27      51. |x|     81. Fx=0
22. П->Х2    52. П->ХС    82. 85
23. X->ПЗ    53. -       83. П->ХД
24. 0        54. Fx>=0   84. С/П
25. X->П2    55. 64      85. П->ХВ
26. КБП7     56. П->Х5    86. П->Х5
27. /-/      57. Вх      87. -
28. X->П2    58. -       88. Fx=0
29. 0        59. Fx>=0   89. 96
```

dump - позволяет вывести содержимое программной памяти в виде последовательности шестнадцатеричных цифр.

```
dump
52 6A 1D 5C 18 63 5E 44 62 0B 43 00 42 87 42 00
43 87 77 63 5E 27 62 43 00 42 87 0B 42 00 43 5D
42 5B 37 6D 50 61 69 12 4C 40 64 62 10 44 65 63
10 45 10 31 6C 11 59 64 65 0F 11 59 63 6D 50 27
65 0F 01 11 11 5E 78 66 64 11 59 78 6B 45 65 6E
11 5E 85 6D 50 6B 65 11 5E 96 05 64 11 5E 96 50
64 A8 65 0E 0E 0E 0E 0E 0E
```


poke [имя регистра стека] [значение] - помещает указанное значение в указанный регистр стека. Имя регистра, указывается в виде одной буквы (X, Y, Z, T). Значение указывается в нормализованном формате, с указанием порядка числа, через латинскую букву "e".

```
poke X 1.25e02
31h 1,32h 2,35h 5,
value '12500000' mpow 1
30h 0,32h 2,
value 12500000 pow 2

poke Y 3.14e02
33h 3,31h 1,34h 4,
value '31400000' mpow 1
30h 0,32h 2,
value 31400000 pow 2
```

kbd [код команды] - эмулирует нажатие на клавиатуры комбинации клавиш с указанным скан-кодом. **[код команды]** указывается в шестнадцатеричном формате и не может превышать 27h. Скан-коды клавиш можно увидеть на приложенной картинке.

<small>MENU</small> 27 ESC	22 ←	1D OK	18 →	13 [USER]	E P	9 ГРД	4 Г
F 26	<small>X<0</small> 21 ШГ→	<small>L0</small> 1C П→X	<small>sin⁻¹ {x}</small> 17 7	<small>cos⁻¹ {x}</small> 12 8	<small>tg⁻¹ max</small> D 9	<small>√</small> 8 —	<small>1/x</small> 3 ÷
25K	<small>X=0</small> 20 ←ШГ	<small>L1</small> 1B X→П	<small>sin⁻¹ x </small> 16 4	<small>cos⁻¹ 3H</small> 11 5	<small>tg⁻¹ δ⁺</small> C 6	<small>π δ⁺</small> 7 +	<small>X²</small> 2 ×
24 ↑↓	<small>X≥0</small> 1F В/О	<small>L2</small> 1A БП	<small>e^x</small> 15 1	<small>lg</small> 10 2	<small>ln δ⁺</small> B 3	<small>x^y δ⁺</small> 6 ↔	<small>Bx δ⁺ CЧ</small> 1 e В↑
23 A↑	<small>X≠0</small> 1E С/П	<small>L3</small> 19 ПП	<small>10^x НОП</small> 14 0	<small>↻</small> F a ·	<small>Λ АВТ V</small> A b /-/	<small>ПРГ ⊕</small> 5 c ВП	<small>CF ИНВ</small> 0 d Cx

Однако нужно иметь ввиду, что при попытке вызова меню, командой терминала (kbd 27), терминал становится на паузу и пока вы не выйдете из сервисного меню на устройстве, продолжить работу с терминалом не получится.

Пример комбинации действий терминала.

```
poke X 1.25e02
poke Y 3.14e02
kbd 2
stk
```

```
poke Y 3.14e02
33h 3,31h 1,34h 4,
value '31400000' mrow 1
30h 0,32h 2,
value 31400000 pow 2
kbd 2
stk
X1 = 1.2500000 02
T = 0.0000000 00
Z = 0.0000000 00
Y = 0.0000000 00
X = 3.9250000 04
IP = 0
```

Вводим в регистр стека X значение 125. Вводим в регистр стека Y значение 314. Имитируем нажатие клавиши [умножить]. Просматриваем результат в регистре стека X. Результат этих действий также отображается на дисплее устройства в виде цифры 39250.

smap (Slot Map) - для быстрой оценки занятости SPI FLASH памяти и поиска свободных слотов, отображает матрицу слотов в формате 10x10, с указанием занятых.

```
smap
  0  1  2  3  4  5  6  7  8  9
0 - [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
1 - [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
2 - [ ][ ][X][ ][ ][ ][ ][ ][ ][ ]
3 - [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
4 - [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
5 - [ ][ ][ ][ ][ ][ ][ ][X][ ][ ][ ]
6 - [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
7 - [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
8 - [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
9 - [ ][ ][ ][ ][ ][ ][ ][ ][ ][ ][ ]
```

snm [номер слота] [имя слота] - присваивает слоту, в SPI FLASH памяти, с указанным номером, указанное имя. Номер слота - двузначное десятичное число. Имя слота - строка состоящая из латинских букв, цифр или символов, длиной не более 14 символов.

Пример:

```
sname 01 BE-HAPPY
sname 22 FISHING
```

sdir (Slot Directory) - отображает список занятых слотов SPI FLASH памяти, с его имени.

```
sdir
22. FISHING
56. BR%ED-2
```

save [номер слота] - сохранение программы, из памяти МК-61s-mini в указанный слот, внешней SPI FLASH памяти. Номер слота - двузначное десятичное число. Если слот уже занят другой программой, то спрашивает подтверждение действия - Y / N.

```
save 43
Enter Y/y to confirm the operation!
Y
```

load [номер слота] - чтение программы в память МК-61s-mini из указанного слота, внешней SPI FLASH памяти. Номер слота - двузначное десятичное число. Если указанный слот пустой, то сообщает об этом звуковым сигналом.

sera (Slots Erase) - Удаляет содержимое всех слотов, в SPI FLASH памяти, помечает их как свободные и стирает их имена . Перед удалением, спрашивает подтверждение действия - Y / N.

sdel [номер слота] (Slot Delete) - помечает указанный слот, в SPI FLASH памяти, незанятым и удаляет имя слота. Номер слота - двузначное десятичное число. Перед удалением, спрашивает подтверждение действия - Y / N. Если указанный слот пустой, то сообщает об этом.

```
sdel 43

Clear slot #43
Enter Y/y to confirm the operation!
Y
```

run - запуск выполнения программы на устройстве. Аналогично последовательности нажатия клавиш F АВТ, В/О, С/П. По окончании выполнения программы, в терминал выдается соответствующее сообщение.

exec - шаг по программе МК-61s-mini в режиме отладки - эквивалентно нажатию клавиши ПП

clr - очищает программную память. Действие требует подтверждения со стороны пользователя, отправкой символа "Y". Отменить данную операцию не представляется возможным.

rst - перезагрузка микроконтроллера

dfu - переход в режим загрузки микроконтроллера по DFU (Device Firmware Uploader или Device Firmware Upgrade).

disa - включение/выключение режима дизассемблера, во всех режимах работы калькулятора, а не только в режиме ввода программы ПРГ