FRANK's SPRITE GENERATOR

Frank Linder

CONTENTS

THE FRANK SPRITE GENERATOR

Begin with typing in to the cursor prompt from a clean start: `LRUN flp1_boot or LRUN mdv1_boot`

F1 - Demo (test) program demonstrates some of the commands in the sprite-generator

F2 - Helicopter, a demonstration game however please note this is not a complete game

F3 - The Picture (Icon) editor is used to make icons, which is used to construct sprites, background or free icons

F4 - The Music editor is used to make small melodies

F5 - The Picture File editor is used to copy icon files and to control icon files.

Q - The Picture Take editor is used to take icons from a saved QL-screen, this is the only program, which works in MODE 4

FILES INCLUDED IN THE PACKAGE

PROGRAMS

- Boot – a boot for all programs
- Ed_boot - a boot for the icon editor
- Heli_boot - a boot for the helicopter game
- Play_boot - a boot for the music editor
- Fil_ed_boot - a boot for the file editor
- Take_boot - a boot for the take editor
- Fsg_tst_boot - a boot for the demo program
- Fsg_v011_cde - the sprite generator
- Copy - a simple copying program

DOCUMENTS

- SPgen_doc - Sprite generator document part 1
- SPgen2_doc - Sprite generator document part 2
- SPgen3_doc - Sprite generator document part 3
- SPgen4_doc - Sprite generator document part 4
- Filed_doc - File editor documentation

- Take_doc - Picture Take editor documentation
- PLAYed_doc - Music editor documentation
- example_doc - Example documentation

ICON-FILES

- Heli_sgs
- Bilboll_sgs
- Bakgrund_sgs
- Aste_sgs
- Grott_sgs

CONTEXT

This sprite generator is meant to deal with:

20 (or more) sprites

1023 'icons' per icon-file

32 'character screens' (only character numbers 1-255 can be used) per screen-file (#)

The sprite generator is installed with:

```
10 cde=RESPR(17000)
20 LBYTES FLP1_FSG_v011_cde, cde
30 CALL cde
```

Throughout this manual I have used these meanings:

icon = p, these are icons created with FSG_editor_bas

character = tkn, icons that have a common size

character-screen = tkns, a whole screen of characters

icon sequence = icons, that arrive in one sequence and of the same size (max 250)

Please Note

sprite = sp, these are moveable things which consists of one or more icons

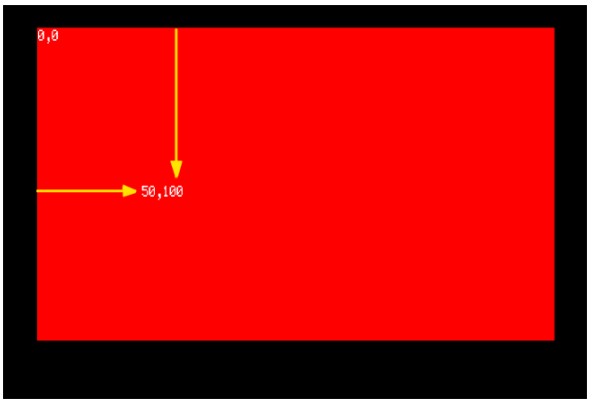
Icons and sprites are XOR'ed on the screen. This means that if the background-colour <> from the one chosen in the editor, the colour of the sprite will differ from the 'original' colours.

The position of an icon or sprite is coordinate to the top left corner of the screen = 0,0. If the position is 50,100 it means that the icon/sprite top left corner is 50 pixels from left border and 100 pixels from top border of the screen.

In order to make icons to sprites, background and so on, load the ICON EDITOR with LRUN flp1_ed_boot.

To make melodies load the PLAY EDITOR with LRUN flp1_PLAYed_boot.

Alternately LRUN flp1_boot and select from the menu.



DEFINING A SPRITE

To define a sprite, you must first:

1: load an icon-file (e.g. spriteicons_sgs) by:

```
b_address=RESPR(5000): LBYTES MDV1_spriteicons_sgs, b_address
```

2: use these three (four) procedures in this order

1. M_PTS
2. CONTROL
3. MOVE_SP
4. (SEQU_SP)

If only MOVE_SP is used the QL will crash, if only one of the two other procedures is used the QL will crash when the function RUN_SP is executed.

When a sprite is defined, M_PTS, CONTROL, MOVE_SP & SEQU_SP can be used freely in your program.

To prevent or detect collisions PT1_SP and PT2_SP is used primarily and BG_SP secondary.

Procedures that (amongst other things) are used for definitions will be marked with ***"-Defining-"***.

Functions marked with ***(*R)*** can only be read after that RUN_SP has been executed.

BEEP should not be used, because that keyword destroys the sound 'timing', use SOUND instead.

Explosion sound is defined with X_NOISE.

If you want sound at all kinds of collisions, you can define that sound (noise) with C_NOISE.

If you have a problem read example_doc or run the DEMO-program.

In addition you can list Fsg_tst_boot, FSG_tst_bas, HELI_boot and HELI_bas.

PROCEDURES

BG_SP sp_nr, test1, test2, breakpoint, end_chr "-Defining-"

BackGround Sprite sets the background sprite and has a total of 5 parameters

sp_nr = sprite number 0 to 19

test1, test2

-1 = stop x and y

0 = nothing

1 = bounce x and y

2 = explosion

3 = stop x

4 = stop y

5 = off remains

6 = off disappear

7 = bounce x

8 = bounce y

9 = invisible

breakpoint

= 1 to 255 (character number)

end chr.

= 1 to 255 must be greater then breakpoint

Test1 tests character 1 until breakpoint, test2 tests the other characters until end_chr, in the last tkns.

Note: P_SCR with three (3) parameters must be used before the background test

CH_HS sp_nr, number

CHange Horizontal Speed - changes the horizontal speed of the sprite and has 2 parameters

sp_nr = sprite number 0 to 19

number = add to the speed

Example: ***CH_HS 3, 10*** *this adds 10 to the horizontal speed of sprite 3*

CH_VS sp_nr, number

CHange Vertical Speed - changes the vertical speed of the sprite and has 2 parameters

sp_nr = sprite number 0 to 19

number = add to the speed

Example: ***CH_VS 4, 12***

adds 12 to the vertical speed of sprite 4

CL_SCR colour

CLear SCReen - clears the screen with the colour 'colour' and has 1 parameter

(background-colour = colour) do not use CLS!

CONTROL sp_nr, on/off, as, ll, rl, ul, bl, ex_time, ex_p, ex_seq "-Defining-"

Control - sets actions and limitations for each sprite and has 10 parameters

sp_nr = sprite number 0 to 19

on/off = 0, 1

As = automatic stop 0, 1, 2, 3, -1

0 = off	
1 = tests the sprite's x position	
2 = tests the sprite's y position	
3 = tests the sprite's x or y position	
-1 = tests the sprite's x and y position	

Auto stop is used when you want a sprite to be inactive (if you want to save time).

ll = left limit 0-255

rl = right limit 0-255

ul = upper limit 0-255

bl = bottom limit 0-255

ex_time = explosion time 0-126 only even numbers

ex_p = explosion points 0-255

ex_seq = explosion sequence speed 0-8

Please Note: The number of explosions icons in a sequence is dependent of ex_time and ex_seq.

See table below, which gives the number of sequence icons.

	ex_time																			
ex_seq	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	3	3	3	3
2	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5
3	1	1	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6	6	7	7
4	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10
6	1	2	2	3	4	4	5	6	6	7	8	8	9	10	10	11	12	12	13	14
8	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

C_NOISE "-Defining-"

Collision NOISE reacts on all types of collisions with 8 parameters (refer to the SuperBASIC BEEP command)

C_N_OFF

Collision Noise OFF

ERASE_S sp_nr [,sp_nr] [,sp_nr] .....

ERASE Sprite erases the sprite and has up to 10 parameters

sp_nr = sprite number 0 to 19

EXPAND "-Defining-"

Expand increases possible number of sprites by 20 and has no parameters

Please Note: Expand may only be executed 1-5 times due to the keyword reserves 3kB after the machine code program (for example if you use EXPAND twice is the number of possible sprites 60). The memory area that reserves for the sprite-generator must be 17000. Add another 3000 for each time EXPAND is executed. EXPAND must be executed direct after the sprite generator is installed.

EXPLODE sp_nr

EXPLODE explodes the sprite and has 1 parameter

sp_nr = sprite number 0 to 19

EMPTY_B

Empty Buffer clears the keyboard-buffer (to be used before INPUT and INKEY$)

F_SLOAD addr, x, y

Franks Screen LOAD copy's a previously saved picture to the screen with the address (addr) and the screen position x, y and has 3 parameters

Addr = Address

x = 0, 4, 8, 12, 16 to 252

y = 0 to 255

F_SSAVE adr, x, y, width, height

Franks Screen SAVE saves a part of screen to memory address (adr) and position x, y and has 5 parameters

x = 0, 4, 8, 12, 16 to 252

y = 0 - 255

width = 4, 8, 12, 16 to 256

height = 1 to 256

You get the address with adr=RESPR(width*height/2+8)

GRAV sp_nr, hor_g, vert_g "-Defining-"

GRAVitation affects the horizontal and vertical axes, applying gravitational features to the sprite

sp_nr = sprite number 0 to 19

hor_g = horizontal gravitation = -120 to 0 to +120

vert_g = vertical gravitation = -120 to 0 to +120

You can experiment with each value in small increments to see the changing effect, alternatively see the demos for usage.

INTSTOP number

INTSTOP is used to provide smoother animation and has 1 parameter

0 = sprites are synchronized with the interrupt

1 = no synchronization

This can affect the flicker of the sprite(s)

I_D_HS sp_nr, number

Increment Decrement Horizontal Speed increases or decreases the speed of the sprite, the speed values are either positive or negative and has 2 parameters.

sp_nr = sprite number 0 to 19

number = -120 to 0 to +120

I_D_VS sp_nr, number

Increment Decrement Vertical Speed increases or decreases the speed of the sprite, the speed values are either positive or negative and has 2 parameters.

sp_nr = sprite number 0 to 19

number = -120 to 0 to +120

JOY1 hsc, max_hs, min_hs, h_grav, hss, vsc, max_vs, min_vs, v_grav, vss, to "-Defining-"

JOYstick 1 accesses SP 0 and detects Arrow Keys and CTRL 1 and has 11 parameters.

NB. See examples section for usage

hsc	= horizontal speed change *64	-1	to 255
max_hs	= maximum horizontal speed	0	to 15
min_hs	= minimum horizontal speed	-15	to 0
h_grav	= horizontal gravitation *64	-100	to +100
hss	= horizontal start speed	-15	to +15
vsc	= vertical speed change *64	-1	to 255
max_vs	= maximum vertical speed	0	to 15
min_vs	= minimum vertical speed	-15	to 0
v_grav	= vertical gravitation *64	-100	to +100
vss	= vertical start speed	-15	to +15
to	= time out for give fire	0	to 30000

Note: Changing of horizontal and vertical speed

1 to 255 = change in speed

-1 = up/down/right/left gives max/min speed

none of these, gives 'start speed' no gravitation

0 = status quo

JOY2 hsc, max_hs, min_hs, h_grav, hss, vsc, max_vs, min_vs, v_grav, vss "-Defining-"
JOYstick 2 accesses SP 1 and detects the five Function Keys and CTRL 2 and has 10 parameters.

NB. See examples section for usage

hsc	= horizontal speed change *64	-1 to 255
max_hs	= maximum horizontal speed	0 to 15
min_hs	= minimum horizontal speed	-15 to 0
h_grav	= horizontal gravitation *64	-100 to +100
hss	= horizontal start speed	-15 to +15
vsc	= vertical speed change *64	-1 to 255
max_vs	= maximum vertical speed	0 to 15
min_vs	= minimum vertical speed	-15 to 0
v_grav	= vertical gravitation *64	-100 to +100
vss	= vertical start speed	-15 to +15

Note: Changing of horizontal and vertical speed

1 to 255 = change in speed

-1 = up/down/right/left gives max/min speed
none of these, gives 'start speed' no gravitation

0 = status quo

LI_MODE sp_nr, lm, rm, um, dm "-Defining-"
LImit MODE defines the screen area in which the sprite can be displayed or travel and has 5 parameters (also see MOVE_SP)

sp_nr	= sprite number 0 to 19	
lm	= left mode	-1 to 6 *(see mode limits)*
rm	= right mode	-1 to 6 *(see mode limits)*
um	= up mode	-1 to 6 *(see mode limits)*
dm	= down mode	-1 to 6 *(see mode limits)*

MOVE_LI sp_nr, x, y "-Defining-"
MOVE LImit moves the limits/borders of sprite sp_nr with x, y steps and has 3 parameters

sp_nr	= sprite number 0 to 19
x	= -200 to 0 to +200
y	= -200 to 0 to +200

MOVE_SP sp_nr, x, y [, hs, vs, lm, rm, um, dm] "-Defining-"
MOVE SPrite moves the sprite sp_nr to position x, y and has 9 parameters

sp_nr = sprite number 0 to +19 visibly
= sprite number -1 to -19 invisibly
x = 0 to 255
y = 0 to 255
hs = horizontal speed -16 to 0 to +16
vs = vertical speed -16 to 0 to +16
lm = left mode -1 to 6 *(see mode limits)*
rm = right mode -1 to 6 *(see mode limits)*
um = up mode -1 to 6 *(see mode limits)*
dm = down mode -1 to 6 *(see mode limits)*
(NB mode = different icons it does not mean it is equal to MODE x as in SuperBASIC)

Mode limits
-1 = 'turn in circle' [1]
0 = stop
1 = bounce [1]
2 = explosion
3 = off ((bouncing) remains)
4 = changes vs/hs and bounce [1]
5 = changes vs/hs and direction [1]
6 = off (disappear)
[1] this does not work together with MOVE_TO and joystick controlled sprites

MOVE_SS sp_nr, sp_nr2 "-Defining-"
MOVE Sprite to Sprite copies sprite data from sp_nr to sp_nr2 and has 2 parameters

sp_nr = sprite number 0 to 19
sp_nr2 = sprite number 0 to 19

MOVE_TO sp_nr, x, y, s, type, hs, vs, ptst
MOVE sprite TO, the sprite sp_nr moves to position x, y with speed s and has either 1 or 8 parameters
If only one parameter is used 'MOVE_TO' is turned off. A sprite which is in the process of a 'MOVE_TO' can't be accessed until the sprite is at the end of the 'MOVE_TO' or if MOVE_TO is used (while it is in the process) with only one parameter the sprite is turned off.

sp_nr = sprite number 0 to 19

x = moves to 0 - 250

y = moves to 0 - 250

s = speed 0 - 15

type = 0 – 4 (see below)

hs = horizontal speed 0 - 15

vs = vertical speed 0 - 15

ptst = point test on/off 0 / 1

Type is used to action the sprite when the sprite arrives at position x, y

0 = stop

1 = off (and remains)

2 = off (and disappears)

3 = explodes

4 = moves with horizontal speed hs and vertical speed vs

M_PTS sp_nr, bap, p1, p2, p3, exp, pv "-Defining-"

Move icons To Sprite defines the icons or sequence of icons the sprite is built from.

sp_nr = sprite number 0 to 19

bap = base_address_icons

p1 = sprite icon number (1) from 1 to 1023

p2 = sprite icon number (2) from 1 to 1023

p3 = sprite icon number (3) from 1 to 1023

exp = explosion icon from 1 to 1023

pv = icon choice 1/2/3/5/6/9/10

See table below

icon choice	tests speed	icon number	results		
0	nothing	1	icon 1		
1	horizontal	2	>0 = icon 1 (p1)	<0 = icon 2 (p2)	
2	vertical	2	*see above*		
5	horizontal	3	>0 = p1	<0 = p2	=0 = p3
6	vertical	3	*see above*		
9	horizontal	3	>1= p1	<-1= p2	= -1 to 1 = p3
10	vertical	3	*see above*		

PRINT_P p_nr, x, y, bap [, ex,p_nr2] [, p_nr3] [,,,,, p_nr8]
PRINT icon 1 to 8 icons are aligned with start position x, y and has between 4 to 6 - 12 parameters

p_nr = icon number 1 to 1023
x = 0 to 250
y = 0 to 250
bap = base_address_icon
ex = 0 – horizontal
= 1 – vertical
p_nr2 = icon number (2) 1 - 1023
p_nr2 = icon number (3) 1 - 1023
……..
p_nr8 = icon number (8) 1 - 1023

P_EXTRA p_nr, x, y, e.nr
Place EXTRA places, one extra icon in the character-screen (there is room for 32 extra icons on each character-screen) and icon p_nr on the VDU and has 4 parameters
NB! P_SCR must be used first.

p_nr = icon number 1 to 1023
x = 0 to 250
y = 0 to 250
e.nr = extra.nr 0 to 31

P_SCR chs_nr, bap, bachs, x_pos, y_pos
Print character SCReen prints a character-screen on the VDU and has between 3 and 5 parameters

chs_nr = chs number 0 to 31
bap = base_address_icon
bachs = base_address_character_screens
x_pos = x position (in chr.) -64 to 64
y_pos = y position (in chr.) -64 to 64

PT1_SP sp_nr, col1, col2, h_mode, v_mode, ti_out, col3 "-Defining-"
Point Test 1 SPrite and has 6 or 7 parameters

sp_nr = sprite number 0 to 19
col1 = colour 0 to 7

col2 = colour 0 to 7

h_mode = horizontal mode -1 to 6 (right/left)

v_mode = vertical mode -1 to 6 (up / down)

mode = modes

- -1 = stop
- 0 = nothing
- 1 = bounce *
- 2 = explosion
- 3 = changes and sets speed positive *
- 4 = changes and sets speed negative *
- 5 = off remains
- 6 = off disappear

* does not work with JOYstick and MOVE_TO

ti_out = time out 1 to 200

col3 = colour 0 to 7

Tests if testpoints diverts from col1, col2 and col3. Test points is set by Icon Editor

PT2_SP sp_nr, col1, col2, h_mode, v_mode, ti_out, col3 "-Defining-"

Point Test 2 SPrite and has 6 or 7 parameters

sp_nr = sprite number 0 to 19

col1 = colour 0 to 7

col2 = colour 0 to 7

h_mode = horizontal mode -1 to 6 (right/left)

v_mode = vertical mode -1 to 6 (up / down)

mode = modes

- -1 = stop
- 0 = nothing
- 1 = bounce *
- 2 = explosion
- 3 = changes and sets speed positive *
- 4 = changes and sets speed negative *
- 5 = off remains
- 6 = off disappear

* does not work with JOYstick and MOVE_TO

ti_out = time out 1 to 200

col3 = colour 0 to 7

Tests if testpoints diverts from col1, col2 and col3. Test points are set by Icon Editor

P_CHR p_nr, x, y, mode

Print CHaracteR prints a character (icon) at character cell x, y in an active character-screen and has 3 or 4 parameters

P_nr = icon number 1 to 255

x = 0 to 250

y = 0 to 250

x = 0 to max (depending on coordinate method)

y = 0 to max (depending on coordinate method)

mode = add the following numbers for chosen combination

0 = character coordinate

1 = pixel coordinate

0 = OVER 0 ('normal print')

2 = replaces a character with a new char with ex_or

0 = saves the char in memory

4 = does not save the char in memory

Mode default = 0

Example **P_CHR 1, 10, 5, 4** Prints Icon 1, 10 across, 5 down on the character coordinate screen, normal print and doesn't save it to memory

Example **P_CHR 1, 10, 5, 5** Prints Icon 1, 10 across, 5 down on the pixel coordinate screen, normal print and doesn't save it to memory

Example **P_CHR 1, 10, 5, 3** Prints Icon 1, 10 across, 5 down on the pixel coordinate screen, replaces the character with ex_or and saves it to memory

Example **P_CHR 1, 10, 5, 7** Prints Icon 1, 10 across, 5 down on the pixel coordinate screen, replaces the character with ex_or and doesn't save it to memory

Mode - Matrix example

Mode value	0	1	2	3	4	5	6	7
Character Coordinate	X		X		X		X	
Pixel Coordinate		X		X		X		X
Normal	X	X			X	X		
Exclusive OR			X	X			X	X
Save	X	X	X	X				
No Save					X	X	X	X

SEQU_SP sp_nr, p_tst, seq.n, seq.spd, seq.mod, md "-Defining-"
SEQUence SPrite and has 6 parameters

NB sequence-icons must have equal size

sp_nr = sprite number 0 to 19

p_tst = point test

0 = at icon 1

1 = at sequence-icons

seq.n = sequence icon number (0) 1 to 250

seq.spd = sequence icon speed -12 to 12

4 = 1 icon / RUN_SP

Seq.mod= sequence mode (see below)

md = missile direction 0/1

1 = depend. sequence icon

0 = depend. first icon

Sequence speed (/4)

Mode	Speed of Sprite	Turn & Return
-9	horizontal (x) *(S)	
-1	vertical (y) *(S)	
0	(S)	
1	(S)	1
2	ABS(horizontal (x)) + ABS((S))	
3	ABS(horizontal (x)) + ABS((S))	1
4	-ABS(horizontal (x)) + ABS((S))	
5	-ABS(horizontal (x)) + ABS((S))	1
10	ABS(vertical (y)) + ABS((S))	
11	ABS(vertical (y)) + ABS((S))	1
12	-ABS(vertical (y)) + ABS((S))	
13	-ABS(vertical (y)) + ABS((S))	1

NB. Sequence speed /4 must not exceed number of icons.

Number of icons ought to be (2*(sequence speed /4))+1.

S_LI sp_nr, max_hs, min_hs, max_vs, min_vs "-Defining-"
Speed LImit specifies the speed limit for the sprite (sp_nr) e.g. minimum and maximum speeds for horizontal and vertical movement and has 5 parameters.

sp_nr	= sprite number 0 to 19	
max_hs	= maximum horizontal speed	-15 to +15
min_hs	= minimum horizontal speed	-15 to +15
max_vs	= maximum vertical speed	-15 to +15
min_vs	= minimum vertical speed	-15 to +15

SPEED sp_nr, hs, vs

Speed specifies the speed for the sprite (sp_nr) e.g. speeds for horizontal and vertical movement and has 3 parameters.

sp_nr	= sprite number 0 to 19	
hs	= horizontal speed	-15 to +15
vs	= vertical speed	-15 to +15

SP_OFF sp_nr [, sp_nr] [, sp_nr] .....

SPeed OFF switches off the sprite (sp_nr) and has up to 10 parameters.

sp_nr = sprite number 0 to 19

NB. This command zeroes the sequence counter

SP_ON sp_nr [, sp_nr] [, sp_nr] .....

SPeed ON switches on the sprite (sp_nr) and has up to 10 parameters.

sp_nr = sprite number 0 to 19

SP_O_E sp_nr [, sp_nr] [, sp_nr] .....

SPrite Off and Erases the sprite (sp_nr) and has up to 10 parameters.

sp_nr = sprite number 0 to 19

SOUND

Equal to BEEP with 8 parameters (refer to the SuperBASIC *BEEP* documentation)

If you give 5 parameters one time and next time only 3 p's, the command will remember the missing two, so you still can realise all 5.

STO_R_V numb, int

STOre Resident Variable saves the integer (numb) and has 2 parameters

numb = 0 to 7

int = 0 to 2^31 (2 000 000 000)

NB. resident even after CLEAR, NEW, LOAD, RUN and LRUN

X_NOISE "-Defining-"

Explosion NOISE with 8 parameters (refer to the SuperBASIC *BEEP* documentation)

X_N_OFF, X_NOISE off

Explosion Noise OFF

Q_CALL

This replaces the QDOS CALL command (refer to the SuperBASIC *CALL* documentation)

FUNCTIONS

A_TST sp_nr (*R)

Automatic stop TeST, tests if the sprite sp_nr is automatically stopped and has 1 parameter

Returns: If automatically stopped 255 else 0

sp_nr = sprite number 0 to 19

CA_TST (*R)

Collision All TeST , tests if there has been any collisions

Returns:	= 0
with a limit	= 1
point test	= 2
background	= 4
explode (direct)	= 8
explosion at a limit	= 16
explosion at point test	= 32
explosion at background	= 64
explosion at MOVE_TO	= 128

If this numb is > than 8 then some sprite has begun to explode and combination reasons are added

Example :-

Limit + explosion at limit	= 17
Point test + explosion at move_to	= 130
Limit + point test	= 3

BG1_TST sp_nr (*R)

BackGround 1 TeST

Returns:	= 0 if collision with sp_nr
sp_nr	= sprite number 0 to 19

BG2_TST

BackGround 2 TeST

Returns:	= 0 if collision with sp_nr

CHR_TST x, y, coordinate, ba_char_s, scr_nr

CharacteR TeST, tests the screen location at x, y depending on coordinate method and has between 2, 3 or 5 parameters

Returns: = character numb

x = 0 to max (depending on coordinate method)

y = 0 to max (depending on coordinate method)

coordinate = 0 character coordinate

= 1 pixel coordinate

ba_char_s = base_address_character screens

scr_nr = character screen = 0 - 31

If the last parameter is not returned, it means that: coordinate = 0 and character-screen is equal to last used character-screen.

DEF_SCR chars_nr, char_width, char_height, ba_char_s

DEFine character SCReen, defines the screen used by characters and is used by the Screen Editor and has 4 parameters

chars_nr = character number = 0 to 31

chars_width = character width = 4 / 8 / 12 / 16 / 24

chars_height = character height = 1 to 24

ba_char_s = base_address_character screens

ENT_TST (*R)

ENTer TeST, returns a value dependant on the pressing of the ENTER key

ESC_TST (*R)

ESC TeST, returns a value of 1 if the ESC key is pressed

EX_SC numb (*R)

EXplosion SCore, returns a value of explosion points and has 1 parameter

numb = 0 function returns the sum of explosion points

= 1 function returns the sum of explosion points and sets points to 0

EX_TST (*R)

EXplosion TeST, tests for any sprite's condition of exploding and returns a value accordingly

Returns: = 255 if any sprite explodes

= 0 if not

EXT_NR ext_nr

EXTra icoN test, tests for extra icon in the active character screen and returns the icon number and has 1 parameter

Returns: = icon number

ext_nr = 0 to 31

EXT_X ext_nr

EXTRA icon test X axis, tests for extra icon in the active character screen and returns it's 'x' axis position and has 1 parameter

Returns: = X-axis position

ext_nr = 0 to 31

EXT_Y ext_nr

EXTRA icon test Y axis, tests for extra icon in the active character screen and returns it's 'y' axis position and has 1 parameter

Returns: = Y-axis position

ext_nr = 0 to 31

JOY_TST (*R)

JOYstick 1 TeST, returns the result of joystick 1 port / arrow-keys and the space key

Returns:	16	8	4	2	1
SPACE (fire)	x				
↑ up-arrow		x			
↓ down-arrow			x		
← left-arrow				x	
→ right-arrow					x

Return example:-

Left arrow + up-arrow = 10

SPACE + up-arrow = 24

SPACE + up-arrow + right arrow = 25

JOY2TST (*R)

JOYstick 2 TeST, returns the result of joystick 2 port / function-keys

Returns:	16	8	4	2	1
F5 (fire)	x				
↑ F4		x			
↓ F2			x		
← F1				x	
→ F3					x

Return example:-

F1 + F4 = 10

F5 + F4 = 24

F5 + F4 + F3 = 25

MISSILE (…, …, …)

MISSILE can be used on a sprite that is either on or off and has between 5, 7 or 8 parameters

When MISSILE is used, the sprite is set on. The sprite cannot be used again with the command MISSILE until the sprite is turned off.

The following keywords close or can close MISSILE:

CONTROL, SP_ON, SP_OFF, EXPLODE

MISSILE is also closed when the sprite has exploded.

A sprite that the MISSILE command has been used, on the other hand, works normally.

5 parameters

sp_nr, sp_nr1, sp_nr2, sp_nr3, sp_nr4

Starts a missile from the sprite sp_nr from the defined 'gun start' with the defined direction

7 parameters

sp_nr, hs, vs, sp_nr1, sp_nr2, sp_nr3, sp_nr4

Starts a missile from the sprite sp_nr from the defined 'gun start' with the speed hs/vs relative the defined direction (NB def.direc. = SGN(def.direc.))

8 parameters

x_pos, y_pos, hs, vs, sp_nr1, sp_nr2 ,sp_nr3, sp_nr4

Equal to the above but the missile starts from x_pos, y_pos with speed hs,vs

First free sprite of sp_nr1 to sp_nr4 is chosen as missile

Returns 0 if no missile is sent away else returns the sprite index (=sp_nrx)

MTO_TST sp_nr (*R)

Move TO TeST, tests sprite (sp_nr) if the Move_to is active and has 1 parameter

Returns: = 255 if MOVE_TO is active

= else 0

OOE_TST sp_nr (*R)

On Off Explosion TeST**,** tests sprite (sp_nr) if the explosion is active and has 1 parameter

Returns: = sprite on, off or exploded

= 255 (on)

= 0 (off)

= 1 to 126 = exploded (counts down)

PLAY pl_adr, tune, tone, time

pl_adr = address to play-file

tune = 0 to 31

tone = 0 to 359

time = -1 to 32000 (beep time)

NB. See PLAYed_doc

P_COL x, y

Point Colour test, tests the pixel colour at point x, y and has 2 parameters

Returns: = colour 0-7

NB. Pixel coordinate for x axis range is 0 to 255

Pixel coordinate for y axis range is 0 to 255

P_TST bg_c_1, bg_c_2, x, y [, x, y] [, x, y] [.....]

Point colour TeST, tests at given coordinates, colours and returns 1 if they are different from bg_c_1 and bg_c_2 and can have 4, 6, 8, 10, or 12 parameters.

Returns: = 1 if coordinate(s) are different from bg_c_1 and bg_c_2

bg_c_1 = background colour 1 = 0 to 7

bg_c_2 = background colour 2 = 0 to 7

x = 0 to 255

y = 0 to 255

PT1_TST sp_nr (*R)

Point Tst 1, tests the sprite (sp_nr) for point value and has 1 parameter

Returns:

Horizontal		Vertical	
test point	*value*	*test point*	*value*
1	1	2	2
3	4	4	8
5	16	6	32
7	64	8	128

PT2_TST sp_nr (*R)

Point Tst 2, tests the sprite (sp_nr) for point value and has 1 parameter

Returns:

Horizontal		Vertical	
test point	*value*	*test point*	*value*
1	1	2	2
3	4	4	8
5	16	6	32
7	64	8	128

RCL_R_V numb

ReCoL Resident Variable returns the saved integer value and has 1 parameter

Returns: = 0 to 2^{32} (4 000 000 000)

RUN_SP (*R)

When this function is executed all the sprites are moving and the sound is started.

Thereafter all the test functions (marked ***(*R)***) can be used.

R_PLAY pl_adr, tune, tone, time

see PLAY

pl_adr = address to play-file

tune = 0 to 31

tone = 0 to 359

time = -1 to 32000 (beep time)

NB. See PLAYed_doc

SEQ_TST sp_nr

SEQuence TeST, returns the sequence-icon-number of the sprite (sp_nr) and has 1 parameter

Returns: = number

NB. If the sprite has 5 sequence icons the function will return 0 to 4

SGN numb

SiGN, returns the sign of a number (numb) and has 1 parameter

Returns: = 1 if positive

= -1 if negative

= 0 if zero

SP_HS sp_nr (*R)

SPrite Horizontal Speed, returns the horizontal speed of sprite (sp_nr) and has 1 parameter

Returns: = number

SP_NX sp_nr (*R)

SPrite Next X, returns with actual speed; 'next X-position' of sprite (sp_nr) and has 1 parameter

Returns: = number

SP_NY sp_nr (*R)

SPrite Next Y, returns with actual speed; 'next Y-position' of sprite (sp_nr) and has 1 parameter

Returns: = number

SP_VS sp_nr (*R)

SPrite Vertical Speed, returns the vertical speed of sprite (sp_nr) and has 1 parameter

Returns: = number

SP_X sp_nr (*R)

SPrite X, returns the horizontal position of sprite (sp_nr) and has 1 parameter

Returns: = x position

SP_Y sp_nr (*R)

SPrite X, returns the vertical position of sprite (sp_nr) and has 1 parameter

Returns: = y position

TIME1 numb (*R)

TIME1, **returns 1 when numb=3 and** each 3:d time that RUN_SP is executed

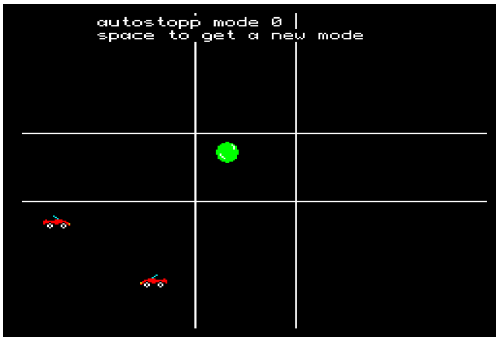
Returns: = value

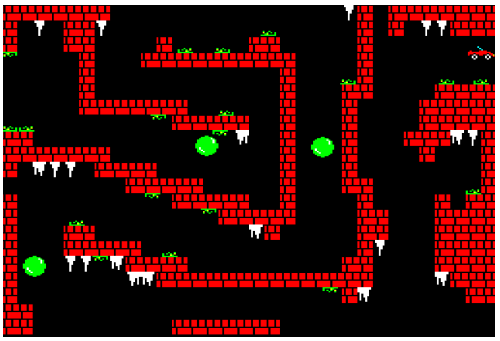
NB. Also TIME2, TIME3 & TIME4

DEMO EXAMPLES (F1 – TEST PROGRAM)

The first option from the main screen menu is F1 - Demo (test) program, this demonstrates some of the commands in the sprite-generator and as can be seen from the image a number of sprite variations are being demonstrated. One of the car sprites are under the control of the keyboard.

The program can be stopped and other demonstration programs can be tested, screen shots below with listing for the v2 demo program.

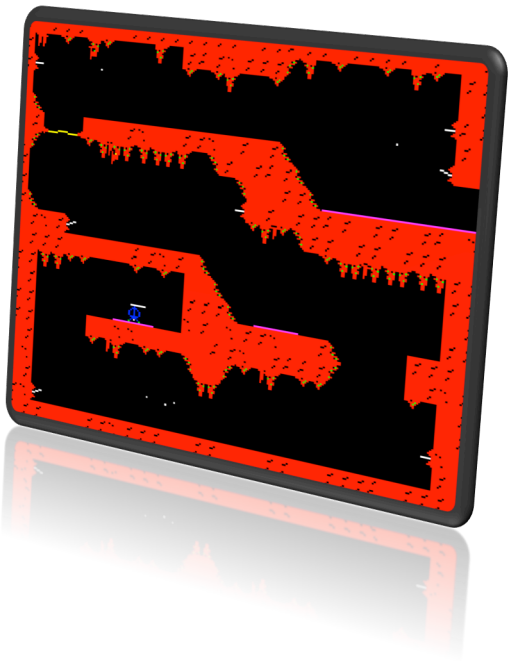




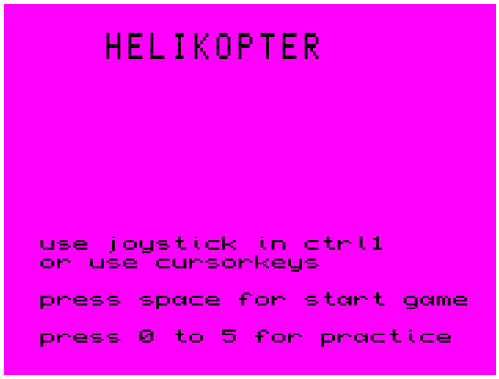

```
2000 DEFine PROCedure v2
2001  SP0
2010  CL_SCR 0:ss 1
2020  SP_OFF 1,2,3
2030  MOVE_SS 0,5:BG_SP 5,9,9,2
2040  MOVE_SP 5,200,180,0,0,0,0,0,0
2050 RESTORE 8000
2060  x5=200:y5=180:h5=3
2230   REPeat game
2260     IF NOT MTO_TST(5)
2270     READ x5,y5,h5
2272      IF x5=-1:RESTORE :READ x5,y5,h5
2280       MOVE_TO 5,x5,y5,h5,0,0,0,0
2290      END IF
2300     bz=RUN_SP
2310     AT 0,0:PRINT MTO_TST(5)
2320   END REPeat game
3050 END DEFine
```

GAME EXAMPLE (F2 - HELIKOPTER GAME)

The second option from the main screen menu is F2 – Helicopter (spelt Helikopter), a demonstration game showing more examples and productive uses of fixed and movable sprites.



It is not a complete game but it does demonstrate game play and allows you to control a sprite with some degree of experiencing real world attributes which includes the cannon option.



Below is a portion of the code from the file 'heli_bas' and show how the additional commands are used, specifically how the Missile command is used, a random number is generated and depending on the random number a missile with defined parameters is generated.

```
10050  defs=RND(1 TO 7)
10060  SELect ON defs
10070   =1:IF de11:mila 225,222,-4,0,0
10080   =2:IF de12:mila 225,42,-4,0,0
10090   =3:a=MISSILE (225,59,-5,-2,15,16,12,14)
10100   =4:a=MISSILE (20,30,4,0,14,16,11,13)
10105   =5:IF de13:a=MISSILE (120,102,-4,0,11,12,13,14)
10106   =6:a=MISSILE (20,232,4,-2,16,15,13,14)
10107   =7:IF de14:a=MISSILE (40,122,4,-2,16,15,13,14)
10110  END SELect
```

ICON EDITOR (F3 - PICTURE EDITOR)

Load the editor with LRUN_FLP1_ED_BOOT

Answer which drive you want to use:

e.g. mdv1_ flp1_

Answer how many icons you want to use.

NB the program reserves number of icons*6.

For 200 sprites 1200 bytes is used. Max 1000 icons.

On top of the command menu is the actual icon number and the length of the sprite file.

NB max length 32000 bytes.

ESC finishes the active command.

F1 - Screen editor

F2 - File editor if the computer is memory-expanded (the icon-file in the icon-editor is the same as file A in the file_editor).

F5 - Zap removes the icon-file

F - Files (SAVE LOAD DELETE DIR)

N - New area changes the area. Change the width first. (Width MOD 4 =0) There after the height.

M - Paint give the colour. To change colour give '\'

S - Big icon enlarges the icon

B - Normal icon draws the icon in normal size.

A - Mirror turns the icon as in a mirror

Q - Upsidedown turns the icon upside down.

K - Mirrorcopy copies the left half mirror vice to the left half.

J - Upsidedowncopy copies the upper half upside down to the bottom half.

O - Save icon puts the icon in the sprite file. NB move towards to the next free space

D - Scroll direct with arrow buttons.

U - New icon moves the first empty space. (According to Y-command)

Y - Change returns to old icon. (If you are editing a loaded file, you can't increase the size)

C - Cls clear active icon.

P - Print picture(s) if more than one icon is to be printed you must give the x/y interval. If the same number is given twice you must as well give the x-pos and y-pos.

T - Test points First question is cannon? cannon = from where a missile can be launched; direction = the direction of a missile; Test points = points tested by PT1_SP and PT2_SP is tested in two groups, up/down and right/left (esc stops)

NB max 8 test points

Used immediately before 'O' command.

R - backgroundcol must be black (0) on all characters.

If a sprite is expected to move on a green background the background colour must be green(4)

NB The real background of the sprite is black

Note the length of the file when you save it, you need it if you want to edit the icon in the editor.

In those computers there '\'-key don't work, use instead the right upper key.

The size of the computer memory, affect the size of the computer icon-file.

FILES

ESC	return to icon editor
N	Change drive flp1_ mdv1_
L	Load \ | enter enables the command
S	Save > | ? gives a directory
D	Delete / |
E	namE
?	Dir
F	File type icons = _sgs / character screen = _tkns
	NB the program adds drive and file suffix
SAVE	overwrites the old file

SCREEN EDITOR

An icon screen is a screen build by several icons (characters). All chars must have the same size. You can have 32 extra icons on the screen. Each extra icon can have any size.

ESC- returns to icon editor

(on top)

number of the iconscreen filesize memory left

total number of char screens latest defined screen

T - Character place the char and press <SPACE>. ESC stops, '\' for new char

If the QL is expanded: "?" will list all characters

C - Cls clears the char-screen with a copy or

with shars

E - Extra icon place the extra icon and press <SPACE>.

ESC stops

N - New extra icon moves the extra icon-pointer

0 - 0 extra icons (removes all extra icons)

D - Define new icon screen NB a char screen can only be defined

once

P - Print char screen

B - change char screen

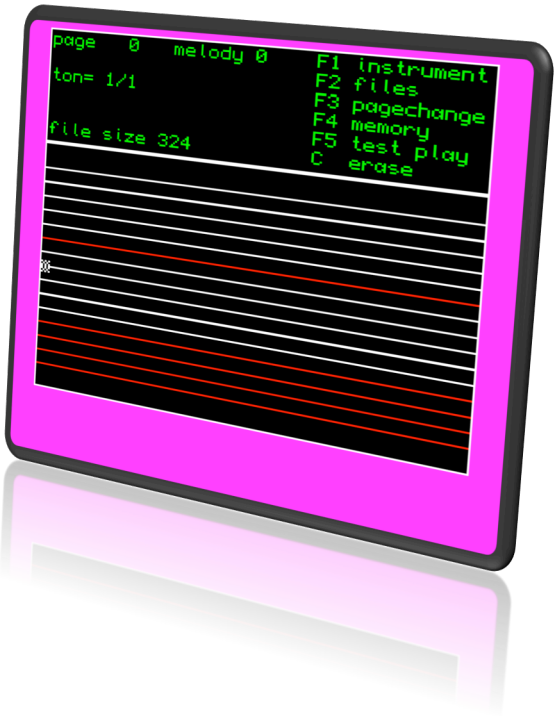
F5 - zap removes all screens

Before you can use an icon screen you must define or LOAD it.

Don't try to use icon, char or icon screen that's not defined.

If the program stops use the command FSG

Play Editor (F4 – Music Editor)



To compose a melody, you do this:

1. enter the notes (max 10 pages)
2. test your tune
3. save the melody to memory
4. note the melody data
5. (eventually a concert)
6. note the file size
7. save the melody on disk with the name 'name' ex. Bach

   The drive name and file-suffix '_plt', is added by the program by default (FLP1_bach_plt).

To give a concert with your melody (melodies) you do like follows:

```
10 Pcde = RESPR(file size)
20 LBYTES FLP1_namn_plt, pcde
30  FOR i=0 to (numb of passes)
40   PAUSE speed * PLAY(pcde, 0, i, (tone length))
50  END FOR i
```

or into a game

```
10 Pcde = RESPR(file size)
20 LBYTES FLP1_namn_plt, pcde
30 a=0:time=0
40  REPeat loop
50   IF TIME1(time)
60    time = timefactor * R_PLAY(pcde,0,a,tone length)
70   END IF
80    IF a>(melody length) THEN a=0
90   fire=RUN_SP
100  game progam......
110 END REPeat loop
```

ICON-FILE EDITOR (F5 – PICTURE FILE EDITOR)

The Icon-File Editor is used to copy icons from one file to another. The program can handle 3 files at the same time, one file (A) copied to and two copied from (B and C)

For each file there is information about:

- file name
- memory space required
- M = file size
- Db = number of defined icons
- Xb = maximum number of icons

information about latest listed icon

icon numb numb bytes

width height

numb test points

cannon definiton (y/n)

F1 - FILES

- S - save file A on ex flp1_
- E - name of file to be saved
- L - load files ? = dir, give file A - C and filenamne
- D - change drive ex flp1_ , mdv1_ ...
- C - H - deletes the whole file A and defines a new file
- S - deletes the last icons from file A

ESC interrupts

The drive name and file-suffix '_sgs' is added by the program

F2 - 'LIST' ICONS

Give icon file A - C and start icon.

← → to list up or down

T - lists test points 0 to 3

U - lists test points 4-7

K - lists on top cannon start, in the bottom direction

NB control how many test points and if cannon is defined.

F3 - COPY

Copies icon or icons to file A from file B - C.

Give file (B, C) a start icon and end icon.

Icon file editor is supposed to use icons which are edited in the icon editor.

If the program stops, use ED1 (FSG)

NB control file size before you load a new file!

TAKE EDITOR (Q – PICTURE TAKE EDITOR)

1. Load the editor with LRUN_FLP1_TAKE_BOOT
2. Answer which drive you want to use: example mdv1_ or flp1_
3. Answer how many icons you want to use. NB the program reserves number of icons*6. For 200 sprites 1200 bytes is used. Max 1000 icons.
4. On top of the command menu is the actual icon number and the length of the sprite file. NB max length 32000 bytes.

In-commands

ESC finishes an active command.

F5 - zap removes the icon-file

F - Files (SAVE LOAD DELETE DIR)

N - New area changes the area. Change the width first (Width MOD 4 =0), thereafter the height.

U - New icon moves the first empty space (According to Y-command).

Y - Change returns to old icon. (If you are editing a loaded file, you can't increase the size)

C - Cls clear active icon.

P - Print picture(s)

Answer the question 'what number ?' with a icon number.

Cursor-keys controls the icon.

Space 'prints' the icon.

T - Take icon(s)

Takes an icon from a screen.

Cursor-keys controls the cursor and the size.

CTRL makes the cursor goes faster.

To change the size use SHIFT.

4 - mode 4 + CLS

8 - mode 8 + CLS

R - backgroundcol must be black (0) on all characters.

If a sprite is expected to move on a green background the background colour must be green(4)

NB The real background of the sprite is black

L - List icons

Answer the question 'what number?' with an icon number.

W - recolour

Note the length of the file when you save it, you need it if you want to edit the icon in the editor. In those computers there '\' - key don't work, use instead the right upper key. The size of the computer memory will affect the size of the computer icon-file.

FILES

ESC- return to icon editor

N - Change drive flp1_ mdv1_

L - Load \ | enter enables the command

S - Save > | ? gives a directory

D - Delete / |

? - Dir

E - Name of file

K - Load screen

P - Save screen

Both save commands overwrites the old file.

RECOLOUR

X - XOR with background colour

R – Recolour

First you will be asked for witch colour you want to change, answer 0 - 7. Secondly you must answer the question 'change to?' answer 0 - 7.

If the program stops use the command FSG

EXAMPLES

```
10 icon_a = RESPR(10000)
```

Reserves 10000 bytes in memory

```
20 LBYTES FLP1_icons_sgs,icon_a
```

Reads 'icons_sgs' to start address icon_a

```
30 M_PTS 0,icon_a, 5, 1, 1, 1, 0
```

Sprite 0 is given icon 5, icon-choice is 0

```
40 M_PTS 4,icon_a, 6, 7, 8, 10, 5
```

Sprite 4 is given 'right icon' 6, 'left icon' 7, 'center icon' 8, explosion icon 10, icon-choice 5

```
50 CONTROL 0,1,0,50,200,0,100,10,0,0
```

Sprite 0 starts up, no automatic stop, left limit = 50, right limit = 200, upper limit = 0, bottom limit = 100, explosion time = 10, 0 points, no explosion sequence

```
60 CONTROL 4, 0, 0, 0, 255, 0, 255, 34, 0, 4
```

Sprite 4 is not stared, no automatic stop, left limit=0, right limit=255, top limit=0, bottom limit=255 (that is to say; moving all over the screen), explosion time = 34, 0 points, explosion-sequence speed = 4 (= 9 explosion sequence-icons)

```
70 SEQU_SP 4, 0, 4, 2, 0, 0
```

Sprite 4,point test at the first icon, number of sequence icons = 4, sequence-speed = 2 (one icon / two RUN_SP), mode 0, missile direction controls by the first icon

```
PT1_SP 4, 0, 1, -1, 1, 2
```

Sprite 4: black and blue colour is not tested, stopped if left/right points reacts, reflects if up/down points reacts, after reaction a timeout 2 (Col 3 not used)

```
JOY1 8, 2, -2, 2, 1, 0, 1, -3, 0, -1, 10
```

Sprite 0: horizontal velocity change = 8 (= 8/64 = 1/8), '→': maximum velocity to the right = 2, '←': max. velocity Left = -2, gravitation = 1 to the right (= 1/64), start-velocity = 1 to the right, vertical velocity change = 0, '↓': velocity 1 down, '↑' velocity 3 up, gravitation = 0, when '↓' and '↑' is not used velocity is 1 up, time delay of fire ('<SPACE>') = 10

```
JOY1 -1, 2, -2, 0, 0, -1, 1, -1, 0, 0, 0
```

sprite 0:

'→' = velocity 2 (to the right)

'←' = velocity -2 (to the left)

'↑' = velocity 1 (up)

'↓' = velocity -1 (down)

no time delay of fire

```
GRAV 4, 0, 10
```

sprite 4: no horizontal gravitation, vertical gravity = 10 (= each tenth time the velocity is changed by 1 (velocity -3 »» -2 2 »» 3), to limit velocity use the procedure S_LI (speed limit)

```
PRINT_P 12, 34, 156, b_address
```

icon numb 12 that is in the picture file at address b_address, will be (XOR) 'printed' (with OVER -1) at position 34, 156

```
PRINT_P 12, 34, 156, b_address, 0, 13, 14
```

see above

icons numb 13 and 14 will be 'printed' to the right of icon numb 12

```
CL_SCR 0:P_SCR 2, b_address, tkns_address
```

background turns black: char-screen 2 (with icon-data beginning at b_address and tkns-data beginning at tkns_address) will be printed

```
Example EXPAND
      10 cde= RESPR(20000):REMark 17000+3000
      20 LBYTES flp1_FSG_v011_cde,cde
      30 CALL cde
      40 EXPAND
      50 REMark you can now use 40 sprites (0 - 39
```