B-ASIC for Amstrad CPC+
Written by Longshot
Logon System

Introduction

B-ASIC is a set of RSX instructions to manage the extra features of the **CPC +**
RSX instruction signify that each instruction is prefixed by a special char : | on english keyboard and “ù” on french azerty keyboard

Since release 3.0 **B-ASIC** runs only on CPC range with 128 Kb memory

Main instructions can manage colours and sprites, providing an extra bank of **128 sprites** for a total of **144 sprites** (since version 3.4)

B-ASIC doesn't take any byte of the CPC locomotive Basic memory.
(The RSX definition and code for locomative basic are not declared in main memory)
The whole interruption system and RSX managment is modified when B-Asic start.

From a technical point of view, the code of the program BASIC using B-ASIC is copied in a bank, because one bank is always open for the system to store the BASIC code.
So, there are two video pages availables for the user.

Historic

Vo	Date	Evolutions
V1.1	??/04/1991	Published in Amstrad Cent pour Cent (A100%) magazine N°38 (may/june 1991) New instructions INKF, INKS, BORDERP, INKFRVB, INKSRVB, BORDERRVB
V1.2	??/05/1991	Published in A100% N°39 (july/aug 1991) New instructions SPRON,SPROFF,SPRZOOM,SPRXY,SPRTDEF,SPRTON,SPRTOFF, SPRSWAP,SPRPLOT,SPRCOPY,SPRTURNX,SPRTURNY,SPRFILL
V1.3	??/08/1991	Published in A100% N°40 (sept/oct 1991) - Corrected bug with MODE instruction with some roms - Corrected bug on prg > 16k - Update instructions SPRTDEF, SPRTON, SPRTOFF - New instructions SPRTORAM, RAMTOSPR, SPRSAVE, SPRLOAD - New instructions RETARDX, RETARDY, MASQUE, SPLIT
V1.4	??/ ??/1992	Published in A100% N°42 () - B-Asic runs in banks (without RSX managment).
V2.1	??/ ??/1992	- New temporary version with RSX system deviation
V3.1	??/??/1992	- 2nd vidéo page. Corrected little bugs. - New instructions POKEASIC, POKEVR, DEFORGXY, DEFORGSPR - New instructions SPRCATCH, INKCOPY,SPRLINK, SPRUNLINK, ERALINK
V3.2	25/02/2006	- Corrected a bug in table calculation (interrupt function) - Corrected a bug in locomotive firmware (ei before real end interrupt)
V3.3	27/02/2006	- Corrected bug. Basic code copied in banque 5 at B-Asic boot (it was a problem for progr bigger than 16 kb) since 3.1 - New instruction SPRWHERE
V3.4	24/05/2007	- Corrected bug in SPRWHERE (variable setting between 4000 and 7FFF) - Update instruction SPRCOPY (now accepting sprites number from 0 to 143). Sprites 0 to 15 are real sprites. Now there are 128 sprites instead of 64 in previous version : 64 sprites FAST and 64 sprites SLOW (sprites access for sprites 80 to 143 are slowest than FAST sprites access). These additionnal sprites does not take more space than with 64 in the previous release. - New instruction DOKE,address, 16bitsvalue - Removing instructions RAMTOSPR and SPRTORAM, which are now unuseful. (replaced by SPRCOPY) - New instruction SPRTSTEP,n1,...,nx to play the sprite-way step by step (user action). - New instruction DEFKBDSPR to link one or two super-sprites with keyboard (or joystick) keys (automatic or user activable). - Update instruction CATCH : New name SPRCATCH - New instruction SPRCOLL to manage (automatically or user-activable) collisions for up to 8 super-sprites. - Some optimisations

Starting B-Asic

To use B-Asic, it's very easy. You need only two things :
Get the file named ***B-ASIC.BIN***
Insert as first line in your basic program the following line :

10 IF PEEK(&BC0E)<>&CD THEN OUT &7F00,&C1 :LOAD «B-ASIC.BIN »,&C000 :CALL &C000

Colours Instructions Set

INKF,pen,colour

- pen=[0..15] colour=[0..4095]
- Modify colour of a pen
- Example : INKF,0,2303

INKFRVB,pen,red,green,blue

- pen=[0..15] red=[0..15] green=[0..15] blue=[0..15]
- Modify colour for a pen using colour elementaries components
- Example : FOR I=0 TO 15:INKFRVB,0,I,0,0:NEXT I

BORDERP,colour

- colour=[0..4095]
- Modify border's colour.
- Example : BORDERP,543

BORDERRVB,red,green,blue

- red=[0..15] green=[0..15] blue=[0..15]
- Modify border colour using colour elementaries components
- Example : FOR I=0 TO 15:BORDERRVB,0,0,0,I:NEXT I

INKS,pen,colour

- pen=[1..15] colour=[0..4095]
- Modify colour of an hardware sprite pen
- Note : The pen 0 cannot be modified. It's the transparent colour.
- Example : INKS,1,3456

INKSRVB,pen,red,green,blue

- pen=[1..15] red=[0..15] green=[0..15] blue=[0..15]
- Modify colour for an hardware sprite pen using colour elementaries components
- Note : The pen 0 cannot be modified. It's the transparent colour.
- Example : FOR I=0 TO 15:INKFRVB,0,I,0,0:NEXT I

INKCOPY

- Copy the whole colour screen palette to sprite palette.

Sprites Instructions Set

SPROFF,SpriteNumber

- SpriteNumber=[0..15]
- Sprite display is disactivated for specified sprite.
- Example : SPROFF,3

SPRON,SpriteNumber

- SpriteNumber=[0..15]
- Sprite display is activated for specified sprite.
- Example : SPRON,3

SPRZOOM,SpriteNumber,XZoom,YZoom

- SpriteNumber=[0..15] XZoom=[0..3] YZoom=[0..3]
- Define sprite zoom for specified sprite.
- If XZoom or YZoom is zeroed, sprite is not displayed
- Zoom value of 1 is no magnify.
- Note : Sprite must be activated by SPRON instruction to be displayed on screen
- Example : SPRZOOM,2,2,1 define the sprite 2 magnification to 2x horiz. and 1x vertically

SPRXY,SpriteNumber,XPos,YPos

- SpriteNumber=[0..15] XPos=[-256..767] YZoom=[-256..255]
- Define sprite position on screen according a mode 2 definition
- Note : Sprite must be activated by SPRON instruction to be displayed on screen
- Example : SPRXY,2,100,80 define sprite 2 position at 100,80 coordinates

SPRTDEF,SpriteNumber,Address,DurationPer,WaitingPer

- SpriteNumber=[0..15] Address=[0..65535] DurationPer=[0..65535] WaitingPer=[0..65535]
- Periods are defined by period of 0,02 second (e.g. 50 = 1 second)
- Define a path for the specified sprite.
- The path is defined by screen positions, given by a set of x,y coordinates defined in memory.
- Each coordinate is defined on 2 bytes (4 bytes for a screen position) in the memory
- The coordinate must be stored on a little endian method, e.g. the less significant byte in first and the most significant byte in second position in memory. You can use the instruction DOKE to do that easily (example : | DOKE address, x : | DOKE address+2,y : address=address+4
- The last x coordinate must take the value F0F0 (hexa) to indicate the path end definition (| DOKE address,&F0F0)
- DurationPer indicates the time while the sprite will run on the path (at the last position the path start again at the fist position).
- If DurationPer is zeroed, so the duration is infinite.
- WaitingPer indicate the start period which occurs when the path will be activated (see SPRTON and SPRTOFF instructions). Note : A same path can be used by one or more sprites. Example : SPRTDEF,3,&3000,0,100 to define for sprite 3 a path defined in &3000 with an infinite loop of the path after 2 seconds of start delay after SPRTON,3 instruction.

SPRTON,SprNb1,SprNb2,...,SprNbx

- SprNbx=[0..15]
- Start the path simultanously for the specified sprites (vbl synchronized for smooth display).
- Note : An undefined path for the sprite may occur an erratic behaviour of the sprite started.
- Example : SPRTON,1,5,7 start the path for sprites 1, 5 and 7 simultaneously

SPRTOFF,SprNb1,SprNb2,...,SprNbx

- SprNbx=[0..15]
- Stop the path simultaneously for the specified sprites
- Example : SPRTOFF,1,7 stop the path for sprites 1 and 7 simultanously

SPRTSTEP,SprNb1,SprNb2,...,SprNbx

- SprNbx=[0..15]
- Execute the path step by step for the specified sprites.
- Note : An undefined path for the sprite may occur an erratic behaviour of the sprite started.
- Example : SPRTSTEP,1,5,7 plays one path step (defined with SPRTDEF) for sprites 1, 5 and 7 simultaneously

SPRSWAP,SpriteNumber1,SpriteNumber2

- SpriteNumberx=[0..15]
- Exchange the real sprite content of the two specified sprites.
- Note : If SpriteNumber1=SpriteNumber2 it results just a waste of time.
- Example : SPRSWAP,1,5 exchange the graphics data of sprites 1 and 5

SPRPLOT,SpriteNumber,XPos,YPos,SpritePen

- SpriteNumber=[0..15] XPos=[0..15] YPos=[0..15] SpritePen=[0..15]
- Plot a pixel in the specified real sprite at specified coordinates
- Note : SpritePen=0 means the transparent pen for sprite
- Example : SPRPLOT,3,10,9,4 plot a pixel pen 4 at coordinates 10,9 of sprite 3

SPRCOPY,SpriteOrg,SpriteDst

- SpriteOrg=[0..143], SpriteDst=[0..143]
- Copy the content of SpriteOrg to SpriteDst.

Sprites number	Definition
0 to 15	Real sprites
16 to 79	Ram Fast sprites
80 to 143	Ram Slow sprites

- Example : SPRCOPY,1,81 copy the content of sprite 1 (asic) to sprite 81 (ram slow)

SPRTURNX,SpriteNumber

- SpriteNumber=[0..15]
- Flip the specified real sprite on horizontal axis
- Example : SPRTURNX,1

SPRTURNY,SpriteNumber

- SpriteNumber=[0..15]
- Flip the specified real sprite on vertical axis
- Example : SPRTURNY,1

SPRFILL,SpriteNumber,SpritePen

- SpriteNumber=[0..15] SpritePen=[0..15]
- Fill the specified sprite with the specified pen
- Note : SpritePen=0 means the transparent pen for sprite
- Example : SPRFILL,1,12 fill the sprite 1 with pen 12

SPRCATCH,SpriteNumber,XPos,YPos

- SpriteNumber=[0..15] XPos=[0..143] YPos=[0..183]
- Get a sprite from mode 0 screen at specified coordinates (coordinates top/left).
- Note : This instruction actually runs only in graphic mode 0
- Example : CATCH,3,120,100 catch the pixels from 120,100 to 135,115 in sprite 3

SPRWHERE,SpriteNumber,@varx%,@vary%

- SpriteNumber=[0..15] varx=Name of x variable vary=Name of y variable
- Get the x,y coordinates of the specified sprite
- Important note : varx and vary must be defined first like integer (declaration in basic with '%') and they must be given to the instruction "by address" ('@' before the variable name). The * declaration must be respected to avoid unpredictibles results.
- Example : 10 VARX%=0: VARY%=0 : SPRWHERE,1,@VARX%,@VARY%: PRINT VARX%,VARY%

SPRSAVE,[SpriteNumber,] BlocNumber, Filename

- SpriteNumber=[0..15] BlocNumber=[0..4] Filename=Name of file in Amsdos standard
- Save a sprite or a set of 32 sprites (defined as one Block) to disk
- B-Asic can store 128 sprites definitions more than the usal sprites contained in asic ram.
- These 128 sprites are organised as 4 sets of 32 sprites named "Block". Each block contains 16 fast sprites and 16 slow sprites

Block & sprites number	Fast Sprites	Slow Sprites
1 – 0 to 15	16 to 31	80 to 95
2 – 0 to 15	32 to 47	96 to 111
3 – 0 to 15	48 to 63	112 to 127
4 – 0 to 15	64 to 79	128 to 143

- Bloc 0 is used for the 16 "real" asic sprites.
- If SpriteNumber is not precised in the instruction, so a complete block is saved
- If SpriteNumber is precised, only two sprites (fast and the slow sprite associated) for the block is saved
- Example : SPRSAVE,3,2,"SPR3BLK1.BIN" save the sprite fast and slow number 3 of the Bloc 2 (e.g. sprite 35 and sprite 99) in the file named "SPR3BLK1.BIN"
- Example : SPRSAVE,0,"SPRASIC.BIN" save the content of block 0 (e.g. 16 asic sprites)

SPRLOAD,[SpriteNumber,] BlocNumber, Filename

- SpriteNumber=[0..15] BlocNumber=[0..4] Filename=Name of file in Amsdos standard
- Load a sprite or a set of 32 sprites (defined as one Block) from disk
- B-Asic can store 128 sprites definitions more than the usal sprite contained in asic ram.
- These 128 sprites are organised as 4 sets of 32 sprites named "Block".

Block & sprites number	Fast Sprites	Slow Sprites
1 – 0 to 15	16 to 31	80 to 95
2 – 0 to 15	32 to 47	96 to 111
3 – 0 to 15	48 to 63	112 to 127
4 – 0 to 15	64 to 79	128 to 143

- Bloc 0 is used for the 16 "real" asic sprites.

- If SpriteNumber is not precised in the instruction, so a complete block is loaded
- If SpriteNumber is precised, only the specified sprite of the block is loaded
- Note : If the file specified has an invalid size, an error is displayed
- Example : SPRLOAD,3,2,"SPR3BLK1.BIN" load the fast and slow sprites 3 of the Bloc 2 (e.g. sprites 35 and 99) from the file "SPR3BLK1.BIN"
- Example : SPRLOAD,3,"RAAHHHHH.SPR" load the block 3 (e.g. sprites 48 to 63 and sprites 112 to 127) from file "RAAHHHHH.SPR"

SPRLINK,SpriteNumber1,SpriteNumber2,Xrel,Yrel

- SpriteNumberx=[0..15] XRel=[-256..767] YRel=[-256..255]
- Link a sprite (2) to another sprite (1) with a relative position. This method can be used to create “super sprites”
- That means if sprite (1) is moved, then sprite (2) is moved too.
- So if sprite 1 is moved in X1,Y1 then the sprite 2 will be moved automatically in X1+Xrel,Y1+Yrel
- It's possible to link all the sprites in all combinations to create one or more "super sprites"
- Note : A sprite cannot be linked to itself.
- Example : SPRLINK,1,2,16,0 link sprite 2 to sprite 1 with a relative x position of 16 pixels
- Example : SPRLINK,2,5,0,16 link sprite 5 to sprite 2 with a relative y position of 16 pixels
- [so move the sprite 1 signify to move sprite 2 and 5]

SPRUNLINK,SpriteNumber1,SpriteNumber2

- SpriteNumberx=[0..15]
- Unlink a sprite from anoter sprite.
- Example : SPRUNLINK,2,5 erase the link between sprite 2 and 5

ERALINK

- Erase all link

SPRCOLL,NumDef,SpriteNum1,DeltaX1,DeltaY1,SpriteNum2,DeltaX2,DeltaY2,@collision%, PixelPerfect
SPRCOLL, NoDef, 255
SPRCOLL

- Process the collision test between two sprites automatically or manually
- NumDef = [0..7] (8 collisions can be processed simultaneously) SpriteNum1=[0..15]
- DeltaXn=[0..255], DeltaYn=[0..255] : Number of pixels in mode 2 definition from sprite coordinate defining the zone to test collision. This method is useful to extend the collision to a super sprite or to restrict collision to only a part of a sprite or supersprite.
- PixelPerfect is not used actually
- @collision% : This variable take the value 1 when a collision is detected between 2 sprites. The user must set the value to 0 to reset the collision detection.

VERY IMPORTANT : The variable collision must be defined like an integer (déclaration by initialisation with %), and the variable must be given to function by address (e.g. @ ou à (on french keyboard) before the variable name).
Example : collision%=0 used in function like @collision%

When SPRCOLL is used with all parameters, the collision process is automatic, and so independant of locomotive basic program. But it's possible to control manually each collision test using the instruction without parameters (at this time, the automatic mode is disactivated). If the user wants to cancel a collision process, he must use the instruction with 2 parameters : SPRCOLL,NumDef,255 (This will help the cpu ;-))

WARNING about SPRCOLL:
SPRCOLL use a locomotive basic variable given by address. And this address is stored by B-Asic when SPRCOLL is called with all parameters. So the user needs ABSOLUTLY keep in mind the two following situation :

- When SPRCOLL is used in automatic mode (e.g B-Asic store automatically the collision test result at each frame), you must absolutly replace the instruction in manual mode (by using SPRCOLL without parameter) before to modify your basic program because the variable is stored and moved in the basic area.
- When you have just stopped your basic program, NEVER modify the program and do a GOTO through a SPRCOLL instruction used without parameters. (the SPRCOLL must be called before with all parameters to give to B-Asic the new address of the variable)

You must respect ABSOLUTLY these 2 rules to avoid a destruction of some part of your basic program. (I will solve the problem of first case later. I need to know when the locomotive basic interpreter is not running. Help appreciate)

DEFORGXY,SpriteNumber,XPos,YPos

- SpriteNumber=[0..15] XPos=[-256..767] YPos=[-256..255]
- Define absolute origin position of the specified sprite
- That means that real sprite position take care of this defined coordinates.
- Example : DEFORGXY,1,10,20:SPRXY,1,5,3 display sprite at 15,23
- Note : It's possible to define an origin position for a sprite following a path (see SPRTDEF)
- The default of origin at B-Asic reset is 0,0

DEFORGSPR,SpriteSlave,SpriteMaster

- SpriteSlave=[0..15] SpriteMaster=[0..15]
- Define the origin position of SpriteSlave with the position of SpriteMaster.
- Note : It's possible to define a sprite relative position for a sprite following a path (see SPRTDEF)
- So, it's possible to generate complex definitions of sprites movments with a supersprite moving relatively to another moving supersprite
- If a sprite 1 use a path 1 and a sprite 2 use a path 2, if the sprite 2 is the slave of sprite 1, then the path 2 will be relative to path 1.
- To erase the slavery of a sprite, use DEFORGXY instruction

DEFKBDSPR, NumDef, SpriteNumber, KeyUp, StepUp, KeyDown, StepDown, KeyLeft, StepLeft, KeyRight, StepRight
DEFKBDSPR,NumDef,255
DEFKBDSPR

- Associate the sprite coordinates to keyboard keys
- NumDef =[0..1] It's possible to define keys for 2 sprites / SpriteNumber=[0..15] Sprite number / Keyxxxxx=[Key number] (see table), Stepxxxx=value 0..255 : Step in mode 2 pixel to add or substract associated value to corresponding key.
- A key 0 indicate to do not test the key. When DEFKBDSPR is defined with all parameters, the sprite process is automatic, and so independent from basic program. It is possible to control manually the sprite movment by using the instruction without parameters (the automatic mode is disactivated).
- Note : If the user wants to cancel a definition, it's better to disactivate the sprite with DEFKBDSPR,NumDef,255 instead of defining all keys with 0 (The B-Asic will gain cpu-speed).

Key Map for DEFKBDSPR :

Numéro	Touche	Numéro	Touche	Numéro	Touche	Numéro	Touche
01	↑	02	→	03	↓	04	F9
05	F6	06	F3	07	ENTER	08	. (pavé n.)
17	←	18	COPY	19	F7	20	F8
21	F5	22	F1	23	F2	24	F0
33	CLR	34	[{	35	RETURN	36	]}
37	F4	38	SHIFT	39	\	40	CTRL
49	£^	50	-=	51	@ |	52	P
53	+ ;	54	* :	55	?/	56	>.
65	_ 0	66	) 9	67	O	68	I
69	L	70	K	71	M	72	< ,
81	(8	82	' 7	83	U	84	Y
85	H	86	J	87	N	88	SPACE
97	&6 J2↑	98	%5 J2↓	99	R J2←	100	T J2→
101	G J2⌖	102	F	103	B	104	V
113	$4	114	#3	115	E	116	W
117	S	118	D	119	C	120	X
129	!1	130	"2	131	ESC	132	Q
133	TAB	134	A	135	CAPS	136	Z
145	J1↑	146	J1↓	147	J1←	148	J1→
149	J1⌖	152	DEL				

Others Instructions Set

POKEASIC,Address,val1,...,valx

- Address=[&4000..&7FFF] Valx=[&00..&FF]
- Poke one or more bytes in Asic I/O page
- Instruction reserved to experimented users.

POKEVR,Address,val1,...,valn

- Address=[&4000..&7FFF] Valx=[&00..&FF]
- Poke one or more bytes in second video page
- Instruction reserved to experimented users.

DOKE,Address,value16bits

- Address=[&4000..&7FFF] value16bits=[&0000..&FFFF]
- Double Poke in user memory

SCREENCOPY,direction

- direction=[0..1]
- 0 : Copy the main videoram to videoram 2
- 1 : Copy the videoram2 to main videoram

Error Messages

Bad arguments number

- Bad number of arguments used in the instruction

Invalid Argument

- An invalid value is used in one or more arguments of the instruction

Invalid Filename

- A bad filename was used in a SPRSAVE or SPRLOAD instruction

Disc Error

- An I/O error has occured in a SPRSAVE or SPRLOAD instruction.
- Disc is missing, or the disk has a wrong or damaged format.

Not a Sprite file

- The size of the file specified in SPRLOAD is not correct.
- See if you do no use the SPRLOAD with a block instead of a double sprite (fast/slow) (or the opposite)

Same Sprites Numbers

- An instruction SPRLINK or SPRUNLINK is attempted with 2 identical sprites numbers.

Bad Graphic Mode

- Instruction SPRCATCH is not used in MODE 0

Invalid Moving List

- An invalid table is created for SPRTDEF
- (e.g. the first coordinate is the path end)

Link Table Full

- The link table is full.
- Normally this error cannot occur in a well built program.

Sprites not Linked

- Instruction SPRUNLINK is used with sprites not linked.

Links & related

- Logon System site for last version : Download B-Asic
- Amstrad Cent pour Cent magazine 38, 39, 40, 42 (sorry only in french)
  http://cpcrulez.free.fr/coding_cpcplus_menu.htm
- Version 3.3 on the cpc "http://www.cpcwiki.com/index.php/B-ASIC"