# Decorative changes

## System font

$3d00 defb …. (Length $0300)

## Copyright message

$1539 defs "© 1982 Sinclair Research Ltd"

Changed to:

$1539 defs "© 1989 DIDAKTIK SKALICA\_\_\_\_\_"

## Sound ‘click’ when key pressed

$1212 LD BC, $0040 ‘for RASP & PIP

Changed to:

$1212 LD BC, $2040

# Well known fixes

## NMI subroutine error

$006d JR NZ, $0070 ‘NO-RESET

Changed to:

$006d JR Z, $0070

I.L.: This routine is not used in the standard Spectrum but the code allows for a system reset to occur following activation of the NMI line. The system variable at 5CB0, named here NMIADD, has to have the value zero for the reset to occur

## 'CURSOR LEFT' subroutine fix

$0a32 LD a, $18

Changed to:

$0a32 LD a, $19

I.L.: Test against top line and accept the change unless at the top of the screen. But this ought to be +19.

## 'CURSOR RIGHT' subroutine fix

$0a4b LD ($5c91), A ‘P-FLAG

RET

Changed to:

$0a4b JP $3226

…

$3226 LD ($5c91), A ‘P\_FLAG

JP $0dd9 ‘CL-SET

I.L.: The programmer has forgotten to exit via PO-STORE (as for ‘Cursor Left’).

P.S.: $3226 is a spare part of changed 'INTEGER TRUNCATION TOWARDS ZERO' subroutine.

## S-SCR-STO' exit fix

$257d JP $2AB2 ‘STK-STO-$

Changed to:

$257d RET

defb $b2, $2a ‘ not used

I.L.: This is jump to stack the matching character, or the null string if no match is found. This exit, via STK-STO-$, is a mistake as it leads to 'double storing' of the string result (see S-STRING, 25DB). The instruction line should be 'RET'.

## 'PRINT A FLOATING-POINT NUMBER' Subroutine fix

$2e24 RST $28 ‘FP-CALC i (i=zero here)

DEFB $E2 ‘get-mem-2 i, f

DEFB $38 ‘end-calc i, f

…

$2e52 POP HL

JP $2ecf ‘PF-FRACTN

Changed to:

$2e24 RST $28 ‘FP-CALC i (i=zero here)

DEFB $02 ‘delete

DEFB $E2 ‘get-mem-2 i, f

DEFB $38 ‘end-calc i, f

‘now all this data have 1 byte offset

…

$2e52 POP HL

JR $2ecf ‘PF-FRACTN

I.L.: the stack is now unbalanced. An extra byte 'DEFB +02, delete' is needed at 2E25, immediately after the RST 0028. Now an expression like "2" +STR$ 0.5 is evaluated incorrectly as 0.5; the zero left on the stack displaces the "2" and is treated as a null string. Similarly all the string comparisons can yield incorrect values if the second string takes the form STR$ x where x is numerically less than 1; e.g. the expression "50"<STR$ 0 .1 yields the logical value "true"; once again "" is used instead of "50".

## 'DIVISION' Subroutine fix

$31ff JR Z, $31E2 ‘DIV-START

Changed to:

$31ff JR Z, $31DB ‘DIV-34TH

I.L.: This jump is made to the wrong place. No 34th bit will ever be obtained without first shifting the dividend. Hence important results like 1/10 and 1/1000 are not rounded up as they should be. Rounding up never occurs when it depends on the 34th bit. The jump should have been to 31DB DIV-34TH above: i.e. byte 3200 hex in the ROM should read DA hex (128 decimal) instead of E1 hex (225 decimal).

## 'INTEGER TRUNCATION TOWARDS ZERO' Subroutine fix

$3223 JR NZ, $323F ‘T-SMALL

INC HL ‘HL is pointed at the fourth byte

INC HL ‘of x, where the 17 bits of the

INC HL

LD A, $80 ‘The first bit is obtained in A.

AND (HL) ‘using 80 hex as a mask.

DEC HL

Changed to:

$3223 JP $323F, T-SMALL

I.L.: The next 26 bytes seem designed to test whether x is in fact -65536 decimal, i.e. 91 80 00 00 00, and if it is, to set it to 00 FF 00 00 00. This is a mistake. As already stated at byte 303B above, the Spectrum system cannot handle this number. The result here is simply to make INT (-65536) return the value -1. This is a pity, since the number would have been perfectly all right if left alone. The remedy would seem to be simply to omit the 28 bytes from 3223 above to 323E inclusive from the program.

## usr-no Subroutine changed

$3331 defb $B3, $34 ‘Offset 2D: 'usr-no'– address $34b3 Changed to:

$3331 defb $aa, $04 ‘new address $04aa

I.L.: This subroutine ('USR number' as distinct from 'USR string') handles the function USR X, where X is a number. The value of X is obtained in BC, a return address is stacked and the machine code is executed from location X.

It is interesting that the IY register pair is re-initialised when the return to STACK-BC has been made, but the important H'L' that holds the next literal pointer is not restored should it have been disturbed. For a successful return to BASIC, H'L' must on exit from the machine code contain the address in SCANNING of the 'end-calc' instruction, 2758 hex (10072 decimal).

$04aa - The following subroutine applies to the ZX81 and was not removed when the program was rewritten for the SPECTRUM.

$04aa CALL $24fb

LD A,($5c3b)

ADD A, A

JP M, $1c8a

POP HL

RET NC

CALL $2bf1

LD H, D

LD L, E

DEC C

RET M

ADD HL, BC

SET 7,(HL)

RET

Changed to:

$04aa CALL $1e99 ‘FIND-INT2

LD HL, $04b6

PUSH HL

LD HL, $2d2b ‘STACK-BC

PUSH BC

RET

$04b6 EXX

LD HL, $2758

EXX

JP $2d2b ‘STACK-BC

DEC DE ‘ not used

SET 7,(HL) ‘ not used

RET ‘ not used

# Original features and fixes

## ‘NEW COMMAND’ changed

$11d0 LD A, $3F

LD I, A

NOP: NOP: NOP

NOP: NOP: NOP ‘wait 24 T states

LD H, D

LD L, E

LD (HL), $02

…

$11ec DEC (HL) ‘02 goes to 01

JR Z, $11eF ‘RAM-DONE - But if zero then RAM is faulty

DEC (HL) ‘01 goes to 00

JR Z, $11e2 ‘RAM-READ

Changed to:

$11d0 JP $397b

$11d3 LD A, $3F

LD I, A

NOP: NOP: NOP ‘wait 12 T states

LD H, D

LD L, E

LD (HL), $FF

…

$11ec INC (HL) ‘ff goes to 00

INC (HL) ‘00 goes to 01

DEC (HL) ‘01 goes to 00

JR Z, $11e2 ‘RAM-READ

NOP

…

New subroutines in spare area:

$397b LD HL, $FFFF ‘wait 10+(6+4+4+12)\*$fffe+(6+4+4+7)=1703941

$397e DEC HL

LD A, H

OR L

JR NZ, $397e

LD A, $92 ‘control ВВ55

OUT ($7F),A

JP $11D3

## ВВ55 Subroutines

New subroutines in spare area:

$386E ld b,a

$386F ld a,($5C80) ‘PR\_CC

and a

jr nz, $3878

$3875 ld a,b

jr $3888

$3878 call $3972

cp l

jr c, $3875

ld a,0D

cp b

jr z, $38C8

call $38C8

jr $386F

$3888 cp $0D

jr z, $38C8

$388C cp $6

jr z, $38D5

$3890 jr $38BC

$3892 ld a, $20

$3894 push af

$3895 ld a, $FE

in a,($FE) ‘ reads keys #FEFE

rra

jr c, $38A6

ld a, $FB

in a,($FE) ‘ reads keys #FBFE

rra

jr c, $38A6

pop af

rst $8 ‘ the ‘error’ restart

ld (de),a

$38A6 in a,($5F) ‘ВВ55 port PC

bit 3,a

jr z, $3895

pop af

ld hl, $5C81 ‘PR\_CC HIGH (NOT\_USED)

bit 7,(hl)

jr z, $38B5

cpl

$38B5 out ($1F),a

ld hl, $5C80 ‘PR\_CC

inc (hl)

ret

$38BC cp $80

jr c, $3894

sub $A5

jr c, $3892

call $0C10 ‘PO-TOKENS Now print the token

ret

$38C8 call $3894

ld a,$0A

call $3894

xor a

ld ($5C80),a ‘PR\_CC

ret

$38D5 ld a,($5C80) ‘PR\_CC

ld b,a

add a,$10

and $F0

sub b

$38DF ld b,a

ld a,20

call $3894

djnz $38DF

ret

$38e7 call $1e85 ‘The two parameters for the OUT instruction

ld b,a ‘are fetched from the calculator stack

in a,($5F)

ld d,a

out (c),b

in a,($5F)

xor d

rra

ret nc

ld hl,($5CB2) ‘set hl = RAMTOP

ld a,h ‘get hi byte

cp $80 ‘check if there are more 16k of RAM available

ret c ‘return if not

di

ld a,c

ld c,$7F

ld ($5C7F),a ‘P\_POSN

ld ($5C3F),sp ‘save sp to LIST SP

ld sp,$8000 ‘set new sp to $8000

ld a,$01

out (c),0

$390e and a

sbc hl,sp

add hl,sp

jr c, $3966

ex (sp),hl

out ($7F),a

ex (sp),hl

out (c),0

ex (sp),hl

pop de

jr $390E

$391E defb $ff … $ff

$3926 jr $3942

call $398A

nop

ld c, $5F

in b,(c)

ld de, $09F4

$3933 out ($7F),a

out (c),b

ld bc, $000F

ld hl,($5C4F) ‘CHANS

add hl,bc

ld (hl),e

inc hl

ld (hl),d

ret

$3942 ld c, $5F

in b,(c)

ld a, $50

ld ($5C81),a ‘PR\_CC HIGH (NOT\_USED)

ld a, $A0

out ($7F),a

ld a,$0D

ld de,$386E

jr $3933

$3956 ld hl, $FFFF

ld bc, $0000

inc hl

ld a, $40

cp h

ret z

ld a, (hl)

xor c

ld c,a

jr $395C

$3966 ld sp,($5C3F) ‘LIST SP

ld a,($5C7F) ‘P\_POSN

ld c,a

out (c),b

ei

ret

$3972 push af

ld a,($5C81) ‘PR\_CC HIGH (NOT\_USED)

res 7,a

ld l,a

pop af

ret

…

$398A ld bc, $5B00 ‘beginning of printer’s buffer

ld ($5C80),bc ‘copy value to PR\_CC

ld a, $92

ret