SPECTRUM 128 KEYPAD SCANNING ROUTINES

MASKABLE INTERRUPT ROUTINE

This routine is nearly identical to that in a standard 48K Spectrum. The only difference is that the normal call to address 02BF to scan the keyboard has been replaced by a call to address 386E. At this address is a new routine which will initiate a scan of the keypad (if in 128K mode) and then the keyboard. Thus compatibility with a 48K Spectrum is maintained although at the minor cost of slightly longer processing time.

```
004A  MASK_INT        PUSH AF
                      PUSH HL

                      LD HL,(5C78) [FRAMES]   The highest byte of the FRAMES
                      INC HL                  counter is only incremented when
                      LD (5C78),HL [FRAMES]   the value of the lower two bytes is
                      LD A,H                  zero
                      OR L
                      JR NZ,0048, KEY_INT

                      INC (IY+40) [FRAMES+3]

0048  KEY_INT         PUSH BC
                      PUSH DE
                      CALL 386E, KEYS         Scan the keypad and the keyboard
                      POP DE
                      POP BC
                      POP HL
                      POP AF
                      EI
                      RET
```

NEW ROM CODE

The new code added to the standard 48K Spectrum ROM is mainly devoted to the scanning and decoding of the keypad. These routines occupy addresses 386E through to 3B3A. Addresses 3B3B through to 3C96 contain a variety of routines for the following purposes: displaying the new tokens ‘PLAY’ and ‘SPECTRUM’, dealing with the keypad when using INKEY$, handling new 128 BASIC error messages, and producing the TV tuner display. Addresses 3BE1 to 3BFE and addresses 3C97 to 3CFF are unused and all contain 00.

SCAN THE KEYPAD AND THE KEYBOARD

This patch will attempt to scan the keypad if in 128K mode and will then scan the keyboard.

```
386E  KEYS            PUSH IX
                      BIT 4,(IY+01) [FLAGS]   Test if in 128K mode
                      JR Z,3879, KEYS_CONT    Z=in 48K mode

                      CALL 3A42, KEYPAD       Attempt to scan the keypad

3879  KEYS_CONT       CALL 02BF, KEYBOARD     Scan the keyboard
                      POP IX
                      RET
```

READ THE STATE OF THE OUTPUT LINES

This routine returns the state of the four output lines (bits 0-3) in the lower four bits of L. The LSB of L corresponds to the output communication line to the keypad. In this way the state of the other three outputs are maintained when the state of the LSB of L is changed and sent out to register 14 of the AY-3-8912.

```
387F  READ_OUTPUTS    LD C,FD                 FFFD = Address of the
```

```
                        LD D,FF         command register (register 7)
                        LD E,BF         BFFD = Address of the
                        LD B,D          data register (register 14)
                        LD A,07
                        OUT (C),A       Select command register
                        IN H,(C)        Read its status
                        LD A,0E
                        OUT (C),A       Select data register
                        IN A,(C)        Read its status
                        OR F0           Mask off the input lines
                        LD L,A          L=state of output lines at the
                        RET             keypad socket
```

SET THE OUTPUT LINE, BIT 0
The output line to the keypad is set via the LSB of L.

```
3896    SET_REG14       LD B,D
                        LD A,0E
                        OUT (C),A       Select the data register
                        LD B,E
                        OUT (C),L       Send L out to the data register
                        RET             Set the output line
```

FETCH THE STATE OF THE INPUT LINE, BIT 5
Return the state of the input line from the keypad in bit 5 of A.

```
389F    GET_REG14       LD B,D
                        LD A,0E
                        OUT (C),A       Select the data register
                        IN A,(C)        Read the input line
                        RET
```

SET THE OUTPUT LINE LOW, BIT 0

```
38A7    RESET_LINE      LD A,L
                        AND FE          Reset bit 0 of L
                        LD L,A
                        JR 3896, SET_REG14  Send out L to the data register
```

SET THE OUTPUT LINE HIGH, BIT 0

```
38AD    SET_LINE        LD A,L
                        OR 01           Set bit 0 of L
                        LD L,A
                        JR 3896, SET_REG14  Send out L to the data register
```

MINOR DELAY ROUTINE
Delay for (B*13)+5 T-States

```
38B3    DELAY           DJNZ 38B3, DELAY
                        RET
```

MAJOR DELAY ROUTINE
Delay for (B*271)+5 T-states

```
38B6    DELAY2          PUSH BC
                        LD B,10
                        CALL 38B3, DELAY    Inner delay of 135 T-States
                        POP BC
                        DJNZ 38B6, DELAY2
```

```
                        RET
```

MONITOR FOR THE INPUT LINE TO GO LOW
Monitor the input line, bit 5, for up to (B*108)+5 T-states.

```
38C0    MON_B5_LO       PUSH BC
                        CALL 389F, GET_REG14        Read the state of the input line
                        POP BC
                        AND 20                      Test bit 5, the input line
                        JR Z,38CB, EXT_MON_LO       Exit if input line found low
                        DJNZ 38C0, MON_B5_LO        Repeat until timeout expires
38CB    EXT_MON_LO      RET
```

MONITOR FOR THE INPUT LINE TO GO HIGH
Monitor the input line, bit 5, for up to (B*108)+5 T-states.

```
38CC    MON_B5_HI       PUSH BC
                        CALL 389F, GET_REG14        Read the state of the input line
                        POP BC
                        AND 20                      Test bit 5, the input line
                        JR NZ,38D7, EXT_MON_HI      Exit if input line found low
                        DJNZ 38CC, MON_B5_HI        Repeat until timeout expires
38D7    EXT_MON_HI      RET
```

READ KEY PRESS STATUS BIT
This entry point is used to read in the status bit for a keypad row. If a key is being pressed in the current row then the bit read in will be a 1.

```
38D8    READ_STATUS     CALL 387F, READ_OUTPUTS     Read the output lines
                        LD B,01                     Read in one bit
                        JR 38E4, READ_BIT
```

READ IN A NIBBLE
This entry point is used to read in a nibble of data from the keypad. It is used for two functions. The first is to read in the poll nibble and the second is to read in a row of key press data. For a nibble of key press data, a bit read in as 1 indicates that the corresponding key was pressed.

```
38DF    READ_NIBBLE     CALL 387F, READ_OUTPUTS     Read the state of the output lines
                        LD B,04                     Read in four bits

38E4    READ_BIT        PUSH BC
                        CALL 389F, GET_REG14        Read the input line from the keypad
                        POP BC
                        AND 20                      This line should initially be high
                        JR Z,392D, LINE_ERROR2      Z=read in a 0, there must be an error

                        XOR A                       The bits read in will be stored in
                                                    register A
38EE    BIT_LOOP        PUSH BC                     Preserve the loop count and any bits
                        PUSH AF                     read in so far
                        CALL 38AD, SET_LINE         Set the output line high

                        LD B,A3                     Monitor for 17609 T-states for the
                        CALL 38C0, MON_B5_LO        input line to go low
                        JR NZ,392B, LINE_ERROR      NZ=the line did not go low

                        CALL 38A7, RESET_LINE       Set the output line low
                        JR 3901, BL_CONTINUE        Insert a delay of 12 T-states
```

```
38FF                    DEFB FF, FF

3901  BL_CONTINUE       LD B,2B                 Delay for 564 T-states
                        CALL 38B3, DELAY
                        CALL 389F, GET_REG14    Read in the bit value
                        BIT 5,A
                        JR Z,3911, BL_READ_0    Z=read in a 0

                        POP AF                  Retrieve read in bits
                        SCF                     Set carry bit
                        JR 3914, BL_STORE

3911  BL_READ_0         POP AF                  Retrieve read in bits
                        SCF
                        CCF                     Clear carry bit

3914  BL_STORE          RRA                     Shift the carry bit into bit 0 of A
                        PUSH AF                 Save bits read in
                        CALL 38AD, SET_LINE     Set the output line high

                        LD B,26                 Delay for 499 T-states
                        CALL 38B3, DELAY

                        CALL 38A7, RESET_LINE   Set the output line low

                        LD B,23                 Delay for 460 T-states
                        CALL 38B3, DELAY

                        POP AF                  Retrieve read in bits
                        POP BC                  Retrieve loop counter and repeat
                        DJNZ 38EE, BIT_LOOP     for all bits to read in
                        RET
```

LINE ERROR

The input line was found at the wrong level. The output line is now set high which will eventually cause the keypad to abandon its transmissions. The upper nibble of system variable FLAGS/ROW3 will be cleared to indicate that communications to the keypad is no longer in progress.

```
392B  LINE_ERROR        POP AF
                        POP BC                  Clear the stack

392D  LINE_ERROR2       CALL 38AD, SET_LINE     Set the output line high

                        XOR A                   Clear FLAGS nibble
                        LD (5B88),A [FLAGS/ROW3]
                        INC A                   Return zero flag reset
                        SCF
                        CCF                     Return carry flag reset
                        RET
```

POLL THE KEYPAD

The Spectrum 128 polls the keypad by changing the state of the output line and monitoring for responses from the keypad on the input line. Before a poll occurs, the poll counter must be decremented until it reaches zero. This counter causes a delay of three seconds before a communications attempt to the keypad is made. The routine can exit at five different places and it is the state of the A register, the zero flag and the carry flag which indicates the cause of the exit. This is summarised below:

A Register	Zero Flag	Carry Flag	Cause
0	set	set	Communications already established
0	set	reset	Nibble read in OK
1	reset	reset	Nibble read in with an error or i/p line initially low
1	reset	set	Poll counter has not yet reached zero

The third bit of the nibble read in must be set for the poll to be subsequently accepted.

```
3938  ATTEMPT_POLL      CALL 387F, READ_OUTPUTS     Read the output line states

                        LD A,(5B88) [FLAGS/ROW3]    Has communications already been
                        AND 80                      established with the keypad?
                        JR NZ,3999, AP_SKIP_POLL    NZ=yes, so skip the poll

                        CALL 389F, GET_REG14        Read the input line
                        AND 20                      It should be high initially
                        JR Z,392D, LINE_ERROR2      Z=error, input line found low

                        LD A,(5B88) [FLAGS/ROW3]    Test if poll counter already zero thus
                        AND A                       indicating a previous comms error
                        JR NZ,395A, POLL_KEYPAD     NZ=ready to poll the keypad

                        INC A                       Indicate comms not established
                        LD (5B88),A [FLAGS/ROW3]
                        LD A,4C                     Reset the poll counter
                        LD (5B89),A [ROW2/ROW1]
                        JR 399C, PK_EXIT            Exit the routine

395A  POLL_KEYPAD       LD A,(5B89) [ROW2/ROW1]     Decrement the poll counter
                        DEC A
                        LD (5B89),A [ROW2/ROW1]
                        JR NZ,399C, PK_EXIT         Exit the routine if it is not yet zero
```

The poll counter has reached zero so a poll of the keypad can now occur.

```
                        XOR A
                        LD (5B88),A [FLAGS/ROW3]    Indicate that a poll can occur
                        LD (5B89),A [ROW2/ROW1]
                        LD (5B8A),A [ROW4/ROW5]     Clear all the row nibble stores

                        CALL 38A7, RESET_LINE       Set the output line low

                        LD B,21                     Wait up to 3569 T-States for the
                        CALL 38C0, MON_B5_LO        input line to go low
                        JR NZ,392D, LINE_ERROR2     NZ=line did not go low

                        CALL 38AD, SET_LINE         Set the output line high

                        LD B,24                     Wait up to 3893 T-States for the
                        CALL 38CC, MON_B5_HI        input line to go high
                        JR Z,392D, LINE_ERROR2      NZ=line did not go high

                        CALL 38A7, RESET_LINE       Set the output line low

                        LD B,0F
                        CALL 38B6, DELAY2           Delay for 4070 T-States
                        CALL 38DF, READ_NIBBLE      Read in a nibble of data
                        JR NZ,392D, LINE_ERROR2     NZ=error occurred when reading
                                                    in nibble
```

```
                        SET 7,A                 Set bit 7
                        AND F0                  Keep only the upper four bits
                                                (Bit 6 will be set if poll successful)
                        LD (5B88),A [FLAGS/ROW3] Store the flags nibble
                        XOR A
                        SRL A                   Exit: Zero flag set, Carry flag reset
                        RET

3999    AP_SKIP_POLL    XOR A                   Communications already established
                        SCF                     Exit: Zero flag set, Carry flag set
                        RET

399C    PK_EXIT         XOR A                   Poll counter not zero
                        INC A
                        SCF                     Exit: Zero flag reset, Carry flag set
                        RET
```

SCAN THE KEYPAD ROUTINE

If a successful poll of the keypad occurs then the five rows of keys are read in and a unique key code generated.

```
39A0    KEYPAD_SCAN     CALL 3938, ATTEMPT_POLL Try to poll the keypad

                        LD A,(5B88) [FLAGS/ROW3] Test the flags nibble
                        CPL
                        AND C0                  Bits 6 and 7 must be set in FLAGS
                        RET NZ                  NZ=poll was not successful
```

The poll was successful so now read in data for the five keypad rows.

```
                        LD IX,5B8A [ROW4/ROW5]
                        LD B,05                 The five rows

39B0    KS_LOOP         PUSH BC                 Save counter

                        CALL 38D8, READ_STATUS  Read the key press status bit
                        JP NZ,3A3A, KS_ERROR    NZ=error occurred

                        BIT 7,A                 Test the bit read in
                        JR Z,39DC, KS_NEXT      Z=no key pressed in this row

                        CALL 38DF, READ_NIBBLE  Read in the row's nibble of data
                        JR NZ,3A3A, KS_ERROR    NZ=error occurred

                        POP BC                  Fetch the nibble loop counter
                        PUSH BC
                        LD C,A                  Move the nibble read in to C
                        LD A,(IX+0)             Fetch the nibble store
                        BIT 0,B                 Test if an upper or lower nibble
                        JR Z,39D6, KS_UPPER     Z=upper nibble

                        SRL C                   Shift the nibble to the lower position
                        SRL C
                        SRL C
                        SRL C
                        AND F0                  Mask off the lower nibble of the
                        JR 39D8, KS_STORE       nibble store

39D6    KS_UPPER        AND 0F                  Mask off the upper nibble of the
                                                nibble store
```

39D8	KS_STORE	OR C	Combine the existing and new
		LD (IX+0),A	nibbles and store them
39DC	KS_NEXT	POP BC	Retrieve the row counter
		BIT 0,B	Test if next nibble store is required
		JR NZ,39E3, KS_NEW	NZ=use same nibble store
		DEC IX	Point to the next nibble store
39E3	KS_NEW	DJNZ 39B0, KS_LOOP	Repeat for the next keypad row

All five rows have now been read so compose a unique code for the key pressed.

		LD E,80	Signal no key press found yet
		LD IX,5B88 [FLAGS/ROW3]	
		LD HL,3A3F, KEY_MASKS	Point to the key mask data
		LD B,03	Scan three nibbles
39F0	GEN_LOOP	LD A,(IX+0)	Fetch a pair of nibbles
		AND (HL)	This will mask off the FLAGS nibble and the SHIFT/0 key
		JR Z,3A17, GEN_NEXT	Z=no key pressed in these nibbles
		BIT 7,E	Test if a key has already been found
		JR Z,3A3C, GEN_INVALID	Z=multiple keys pressed
		PUSH BC	Save the loop counter
		PUSH AF	Save the byte of key bit data
		LD A,B	Move loop counter to A
		JR 3A01, GEN_CONT	A delay of 12 T-States
39FF		DEFB FF, FF	
3A01	GEN_CONT	DEC A	These lines of code generate base
		SLA A	values of 7, 15 and 23 for the three
		SLA A	nibble stores 5B88, 5B89 & 5B8A.
		SLA A	
		OR 07	
		LD B,A	B=(loop counter-1)*8+7
		POP AF	Fetch the byte of key press data
3A0C	GEN_BIT	SLA A	Shift until a set key bit drops into the
		JP C,3A13, GEN_FOUND	carry flag
		DJNZ 3A0C, GEN_BIT	Decrement B for each ‘unsuccessful’ shift of the A register
3A13	GEN_FOUND	LD E,B	E=a unique number for the key pressed, between 1 - 19 except 2 & 3
		POP BC	As a result shifting the set key bit into the carry flag, the A register will hold 00 if only one key was pressed
		JR NZ,3A3C, GEN_INVALID	NZ=multiple keys pressed
3A17	GEN_NEXT	INC IX	Point to the next nibble store
		INC HL	Point to the corresponding mask data
		DJNZ 39F0, GEN_LOOP	Repeat for all three ‘nibble’ bytes
		BIT 7,E	Test if any keys were pressed

```
                        JR NZ,3A27, GEN_POINT       NZ=no keys were pressed

                        LD A,E                      Copy the key code
                        AND FC                      Test for the '.' key (E=1)
                        JR Z,3A27, GEN_POINT        Z='.' key pressed

                        DEC E
                        DEC E                       Key code in range 2 - 17
```

The E register now holds a unique key code value between 1 and 17.

```
3A27    GEN_POINT       LD A,(5B8A) [ROW4/ROW5]     Test if the SHIFT key was pressed
                        AND 08
                        JR Z,3A34, GEN_NOSHIFT      Z=the SHIFT key was not pressed
```

The SHIFT key was pressed or no key was pressed.

```
                        LD A,E                      Fetch the key code
                        AND 7F                      Mask off 'no key pressed' bit
                        ADD A,12                    Add on a shift offset of 12
                        LD E,A
```

Add a base offset of 5A to all key codes. Note that no key press will result in a key code of DA. This is the only code with bit 7 set and so will be detected later.

```
3A34    GEN_NOSHIFT     LD A,E
                        ADD A,5A                    Add a base offset of 5A
                        LD E,A                      Return key codes in range 5B - 7D
                        XOR A
                        RET                         Exit: Zero flag set, key found OK
```

These two lines belong with the loop above to read in the five keypad rows and are jumped to when an error occurs during reading in a nibble of data.

```
3A3A    KS_ERROR        POP BC                      Clear the stack and exit
                        RET                         Exit: Zero flag reset

3A3C    GEN_INVALID     XOR A                       Exit: Zero flag reset indicating an
                        INC A                       invalid key press
                        RET
```

KEYPAD MASK DATA

```
3A3F    KEY_MASKS       DEFB 0F, FF, F2             Key mask data
```

READ THE KEYPAD

This routine reads the keypad and handles key repeat and decoding. The bulk of the key repeat code is very similar to that used in the equivalent keyboard routine and works are follows. A double system of KSTATE system variables (KSTATE0 - KSTATE3 and KSTATE4 - KSTATE7) is used to allow the detection of one key while in the repeat period of the previous key. In this way, a 'spike' from another key will not stop the previous key from repeating. For a new key to be acknowledged, it must be held down for at least 1/5th of a second, i.e. ten calls to KEYPAD. The KSTATE system variables store the following data:

KSTATE0/4	Un-decoded Key Value (00-27 for keyboard, 5B-7D for keypad, FF for no key)
KSTATE1/5	10 Call Counter
KSTATE2/6	Repeat Delay
KSTATE3/7	Decoded Key Value

The code returned is then stored in system variable LAST_K (5C08) and a new key signalled by setting bit 5 of FLAGS (5C3B).

If the Spectrum 128 were to operate identically to the standard 48K Spectrum when in 48K mode, it would have to spend zero time in reading the keypad. As this is not possible, the loading on the CPU is reduced by scanning the keypad upon every other interrupt. A '10 Call Counter' is then used to ensure that a key is held down for at least 1/5th of a second before it is registered. Note that this is twice as long as for keyboard key presses and so the keypad key repeat delay is halved.

At every other interrupt the keypad scanning routine is skipped. The net result of the routine is simply to decrement both '10 Call Counters', if appropriate. By loading the E register with 80 ensures that the call to KP_TEST will reject the key code and cause a return. A test for keyboard key codes prevents the Call Counter decrements affecting a keyboard key press. It would have been more efficient to execute a return upon every other call to KEYPAD and then to have used a '5 Call Counter' just as the keyboard routine does.

A side effect of both the keyboard and keypad using the same KSTATE system variables is that if a key is held down on the keypad and then a key is held down on the keyboard, both keys will be monitored and repeated alternatively, but with a reduced repeat delay. This delay is between the keypad key repeat delay and the keyboard key repeat delay. This occurs because both the keypad and keyboard routines will decrement the KSTATE system variable Call Counters. The keypad routine 'knows' of the existence of keyboard key codes but the reverse is not true.

3A42	KEYPAD	LD E,80	Signal no key pressed
		LD A,(5C78) [FRAMES]	
		AND 01	Scan the keypad every other
		JR NZ,3A4F, KP_CHECK	interrupt
		CALL 39A0, KEYPAD_SCAN	
		RET NZ	NZ=no valid key pressed
3A4F	KP_CHECK	LD HL,5C00 [KSTATE0]	Test the first KSTATE variable
3A52	KP_LOOP	BIT 7,(HL)	Is the set free?
		JR NZ,3A62, KP_CH_SET	NZ=yes
		LD A,(HL)	Fetch the un-decoded key value
		CP 5B	Is it a keyboard code?
		JR C,3A62, KP_CH_SET	C=yes, so do not decrement counter
		INC HL	
		DEC (HL)	Decrement the 10 Call Counter
		DEC HL	
		JR NZ,3A62, KP_CH_SET	If the counter reaches zero, then signal the set is free
		LD (HL),FF	
3A62	KP_CH_SET	LD A,L	Jump back and test the second set if
		LD HL,5C04 [KSTATE4]	not yet considered
		CP L	
		JR NZ,3A52 KP_LOOP	
		CALL 3AAE, KP_TEST	Test for valid key combinations and
		RET NZ	return if invalid
		LD A,E	Test if the key in the first set is being
		LD HL,5C00 [KSTATE0]	repeated
		CP (HL)	
		JR Z,3A9E, KP_REPEAT	Jump if being repeated

```
                        EX DE,HL                Save the address of KSTATE0
                        LD HL,5C04 [KSTATE4]    Test if the key in the second set is
                        CP (HL)                 being repeated
                        JR Z,3A9E, KP_REPEAT    Jump if being repeated
```

A new key will not be accepted unless one of the KSTATE sets is free.

```
                        BIT 7,(HL)              Test if the second set is free
                        JR NZ,3A83, KP_NEW      Jump if set is free

                        EX DE,HL
                        BIT 7,(HL)              Test if the first set is free
                        RET Z                   Return if no set is free

3A83    KP_NEW          LD E,A                  Pass the key code to the E register
                        LD (HL),A               and to KSTATE0/4
                        INC HL
                        LD (HL),0A              Set the '10 Call Counter' to 10
                        INC HL

                        LD A,(5C09) [REPDEL]    Fetch the initial repeat delay
                        SRL A                   Divide delay by two
                        LD (HL),A               Store the repeat delay
                        INC HL

                        CALL 3AD7, KP_DECODE    Decode the keypad key code
                        LD (HL),E               and store it in KSTATE3/7
```

This section is common for both new keys and repeated keys.

```
3A94    KP_END          LD A,E
                        LD (5C08),A [LAST_K]    Store the key value in LAST_K
                        LD HL,5C3B, FLAGS
                        SET 5,(HL)              Signal a new key pressed
                        RET
```

THE KEY REPEAT SUBROUTINE

```
3A9E    KP_REPEAT       INC HL
                        LD (HL),0A              Reset the '10 Call Counter' to 10
                        INC HL
                        DEC (HL)                Decrement the repeat delay
                        RET NZ                  Return if not zero

                        LD A,(5C0A) [REPPER]    The subsequent repeat delay is
                        SRL A                   divided by two and stored
                        LD (HL),A
                        INC HL
                        LD E,(HL)               The key repeating is fetched
                        JR 3A94, KP_END         and then returned in LAST_K
```

THE TEST FOR A VALID KEY CODE SUBROUTINE

The zero flag is returned set if the key code is valid. No key press, SHIFT only or invalid shifted key presses return the zero flag reset.

```
3AAE    KP_TEST         LD A,E
                        LD HL,5B66, FLAGS3      Test if in BASIC or EDIT mode
                        BIT 0,(HL)
                        JR Z,3ABC, KPT_EDIT     Z=EDIT mode
```

Test key codes when in BASIC/CALCULATOR mode

```
                        CP 6D                   Test for shifted keys
                        JR NC,3AD4, KPT_INVALID and signal an error if found

3ABA  KPT_OK            XOR A                   Signal valid key code
                        RET                     Exit: Zero flag set
```

Test key codes when in EDIT/MENU mode.

```
3ABC  KPT_EDIT          CP 80                   Test for no key press
                        JR NC,3AD4, KPT_INVALID NC=no key press

                        CP 6C                   Test for SHIFT on its own
                        JR NZ,3ABA, KPT_OK      NZ=valid key code

3AC4                    DEFB 00, 00, 00, 00     Delay for 64 T-States
                        DEFB 00, 00, 00, 00
                        DEFB 00, 00, 00, 00
                        DEFB 00, 00, 00, 00

3AD4  KPT_INVALID       XOR A                   Signal invalid key code
                        INC A
                        RET                     Exit: Zero flag reset
```

THE KEY DECODING SUBROUTINE

```
3AD7  KP_DECODE         PUSH HL                 Save the KSTATE pointer
                        LD A,E
                        SUB 5B                  Reduce the key code range to
                        LD D,00                 00 - 22 and transfer to DE
                        LD E,A

                        LD HL,5B66, FLAGS3      Test if in EDIT or BASIC mode
                        BIT 0,(HL)
                        JR Z,3AEA, KPD_EDIT     Z=EDIT/MENU mode
```

Use Table 1 when in CALCULATOR/BASIC mode.

```
                        LD HL,3B13, KPD_TABLE1
                        JR 3B0F, KPD_EXIT       Look up the key value
```

Deal with EDIT/MENU mode.

```
3AEA  KPD_EDIT          LD HL,3B25, KPD_TABLE4  Use Table 4 for unshifted key
                        CP 11                   presses
                        JR C,3B0F, KPD_EXIT
```

Deal with shifted keys in EDIT/MENU mode.

Use Table 3 with SHIFT 1 (delete to beginning of line), SHIFT 2 (delete to end of line), SHIFT 3 (SHIFT TOGGLE). Note that although SHIFT TOGGLE produces a unique valid code, it actually performs no function when editing a BASIC program.

```
                        LD HL,3B21, KPD_TABLE3
                        CP 15                   Test for SHIFT 1
                        JR Z,3B0F, KPD_EXIT
```

```
                        CP 16                   Test for SHIFT 2
                        JR Z,3B0F, KPD_EXIT
                        JR 3B01, KPD_CONT       Delay for 12 T-States

3AFE                    DEFB 00, FF, FF         Unused locations

3B01    KPD_CONT        CP 17                   Test for SHIFT 3
                        JR Z,3B0F, KPD_EXIT
```

Use Table 2 with SHIFT 4 (delete to beginning of word) and SHIFT 5 (delete to end of word).

```
                        LD HL,3B18, KPD_TABLE2
                        CP 21                   Test for SHIFT 4 and above
                        JR NC,3B0F, KPD_EXIT
```

Use Table 1 for all other shifted key presses.

```
                        LD HL,3B13, KPD_TABLE1

3B0F    KPD_EXIT        ADD HL,DE               Look up the key value
                        LD E,(HL)
                        POP HL                  Retrieve the KSTATE address
                        RET
```

THE KEYPAD DECODE LOOK-UP TABLES

```
3B13    KPD_TABLE1      DEFB 2E, 0D, 33, 32     '.', ENTER, 3, 2
                        DEFB 31                 1
3B18    KPD_TABLE2      DEFB 29, 28, 2A, 2F     ), (, *, /
                        DEFB 2D, 39, 38, 37     - , 9, 8, 7
                        DEFB 2B                 +
3B21    KPD_TABLE3      DEFB 36, 35, 34, 30     6, 5, 4, 0
3B25    KPD_TABLE4      DEFB A5, 0D, A6, A7     bottom, ENTER, top, end of line
                        DEFB A8, A9, AA, 0B     start of line, TOGGLE, DEL right, Up
                        DEFB 0C, 07, 09, 0A     DEL, CMND, Right, Down
                        DEFB 08, AC, AD, AE     Left, down ten, up ten, end word
                        DEFB AF                 beginning of word
                        DEFB B0, B1, B2, B3     DEL to end of line, DEL to start of
                                                line, SHIFT TOGGLE, DEL to end
                                                of word
                        DEFB B4                 DEL to beginning of word

3B3B – 3B6B                                     Other new Spectrum 128 routines
                                                occupy these locations. They do not
                                                deal with the keypad.
```

INKEY$ ROUTINE TO DEAL WITH THE KEYPAD

```
3B6C    KEYSCAN2        CALL 028E, KEYSCAN      Scan the keyboard
                        LD C,00
                        JR NZ,3B80, KPI_SCAN    NZ=multiple keys

                        CALL 031E, K_TEST
                        JR NC,3B80, KPI_SCAN    NC=shift only or no key

                        DEC D
                        LD E,A
                        CALL 0333, K_DECODE
                        JP 2657, S_CONT         Get string and continue scanning
```

3B80	KPI_SCAN	BIT 4,(IY+01) JP Z,2660, S_IK$_STK DI CALL 39A0, KEYPAD_SCAN EI JR NZ,3B9A, KPI_INVALID CALL 3AAE, KP_TEST JR NZ,3B9A, KPI_ INVALID CALL 3AD7, KP_DECODE LD A,E JP 2657, S_CONT	128K mode? Z=no, stack keyboard code Disable interrupts whilst scanning the keypad NZ=multiple keys Test the keypad key code NZ=no key, shift only or invalid combination Form the key code Get string and continue scanning
3B9A	KPI_ INVALID	LD C,00 JP 2660, S_IK$_STK	Signal no key, i.e. length=0
3B9F – 3BDD			Other new Spectrum 128 routines occupy these locations and these do not deal with the keypad.
3BDE	KP_SCAN2	JP 3C01, KP_SCAN	This is not called from either ROM. It can be used to scan the keypad.
3BE1 – 3C00			Other new Spectrum 128 routines occupy these locations and these do not deal with the keypad.
3C01	KP_SCAN	JP 39A0, KEYPAD_SCAN	This was to be called via the vector table in the EDITOR ROM but due to a programming error it never gets called.
3C04 – 3C96			Other new Spectrum 128 routines occupy these locations and these do not deal with the keypad.
3C97 – 3CFF			Unused locations

EDITOR ROM CODE

The EDITOR ROM does not contain any routines to directly scan or decode the keypad. It does however contain a fifteen entry vector table at location 0100 that points to useful routines within the ROMs, including the keypad scanning routine. The table is designed to allow machine code programs to reliably access these routines even if subsequent versions of the ROMs store them at different locations. Note that a programming error prevents the keypad entry from working.

0100 – 0117			Vector table entries. These do not relate to the keypad.
0118	KPSCAN	JP 012D, KPSCAN2	Vector table entry for the keypad routine.
011B – 012C			Vector table entries. These do not relate to the keypad.
012D	KPSCAN2	RST 28 DEFW 3B01, KP_SCAN RET	Make a 48K ROM call to KP_SCAN. Note that this should have been 3C01.