ZX Spectrum Keyboard Technical Document (prerelease 13/05/2015).

The ZX Spectrum Keyboard has two layers, the Game layer and the QWERTY layer. Each has a specific function and purpose, which will be described here.

1. The Game layer.

This uses a non-standard protocol to communicate, whereby it uses a different keycode to indicate that the key has been released.

For example, if you press the 1 key.

Instead of the usual:
> **Keydown 1.**

Then you release the 1 key:
> **Keyup 1.**

With the Game layer:

You press the 1 key:
> **Keydown a.**
> **Keyup a.**

You release the 1 key:
> **Keydown b.**
> **Keyup b.**

- There are 40 keys overall, each with a different keydown and keyup code.
- There is no auto-repeat.
- Caps Lock has no effect on this layer.

Example Code:

zxkeyboard.c
zxkeyboard.h
WinMain.cpp
resource.h
resource.rc

A Microsoft Visual Studio 2015 project file is included (if you're using an older version of Visual Studio, it's just an empty Win32 project with the 5 files included).

The portable files are zxkeyboard.c and zxkeyboard.h.

```
int ZXKeyboard_ASCIIKeyPress(STRUCT ZXKEYBOARD *zxkb, ZXWCHAR key); // Call this to report a
keypress to the routines.
int *ZXKeyboard_GetMap(STRUCT ZXKEYBOARD *zxkb);                     // Returns array of
keypresses (1 = pressed, 0 = unpressed).
ZXWCHAR *ZXKeyboard_GetMapVisual(STRUCT ZXKEYBOARD *zxkb);           // Get string
representation of current keyboard state (for demo purposes).
void ZXKeyboard_Reset(STRUCT ZXKEYBOARD *zxkb);                      // Reset internal
structures.
void ZXKeyboard_Init(STRUCT ZXKEYBOARD *zxkb);                       // Initialise keyboard
routines.
```

Basically...

```
#include "zxkeyboard.h"
struct ZXKEYBOARD gZXKB;
```

... then initialise...

```
ZXKeyboard_Init(&gZXKB);
```

Report a keypress with:

```
ZXKeyboard_ASCIIKeyPress(&gZXKB, wCharacterCode);
```

Read the keymap like this:

```
int *pKeyMap = ZXKeyboard_GetMap(&gZXKB);
```

"pKeyMap" points to an array of 40 ints, which are set to 1 if pressed or 0 if not pressed.
You can access each key by using one of the enumerated offsets into the array, for example:

```
if(pKeyMap[eZXKEYBOARD_CAPS_SHIFT])
{
      // Caps Shift is currently pressed.
}
```

(All of the enumerated keys are listed in zxkeyboard.h)

And whenever you want to reset the keymap:

```
ZXKeyboard_Reset(&gZXKB);
```

Key up/down list

ZX Spectrum	*Key Press*	*Key Release*	*Key Press names*	*Key Release names*
1	a	b		
2	c	d		
3	e	f		
4	g	h		
5	i	j		
6	k	l		
7	m	n		
8	o	p		
9	q	r		
0	s	t		
Q	u	v		
W	w	x		
E	y	z		
R	A	B		
T	C	D		
Y	E	F		
U	G	H		
I	I	J		
O	K	L		
P	M	N		
A	O	P		
S	Q	R		
D	S	T		
F	U	V		
G	W	X		
H	Y	Z		
J	0	1		
K	2	3		
L	4	5		
ENTER	6	7		
CAPS	8	9		
Z	<	>	Less Than	Greater Than
X	-	"="	Minus	Equal
C	[	]	L Square.	R Square.
V	;	:	Semicolon	Colon
B	,	.	Comma	Dot
N	/	?	Slash	Question Mark
M	{	}	Left Curly	Right Curly
SYMBOL	!	$	Exclamation	Dollar
SPACE	%	^	Percent	Circumf.

2. QWERTY layer.

This uses the standard keydown and keyup codes for 38 of the 40 keys (the alphanumeric keys). However, as there are far fewer keys available than a normal keyboard, the two shift keys are used in combination with other keys to allow the full range of keys to be typed.

This has implications when emulating a ZX Spectrum. You can use the other 38 keys without a problem, but Caps Shift and Symbol Shift will not operate in the way you expect, as they do not generate keycodes by themselves. One obvious example is emulating 48 BASIC. Caps Shift and Symbol Shift put you into "Extended Mode", but you cannot do this with the QWERTY layer.

You can use any game that can be operated without the two shift keys and it will be just fine. However, because of the limitations, it is strongly recommended that developers should add support for the Game layer.

Mac/Windows/Android Layer

Note the entries, in the table below, marked "Mac Layer" / "Windows Layer" / "Android Layer".

Certain keycodes are different on each machine, so these options have been added to allow the keyboard to be used more effectively on these machines.

Certain symbols, e.g. **# " @ \ | ~ £** are different on each machine.

The **Mac** layer will work on all Macs and iOS devices.
The **Windows** layer is configured for a UK Windows keyboard.
The **Android** layer is a compromise. We found that different Android devices behaved differently, with respect to keyboard codes, so you may find that setting your device to either UK layout or US layout is better. Alternatively, you may find that setting the keyboard layout makes no difference! However, with the best configuration you should be able to get all of the codes with the exception of the pound (£) key.

Please note that these layer-switching keys need to be held down for 1 second to take effect.

Mac layer: Caps Shift + Symbol Shift + Z
Windows layer: Caps Shift + Symbol Shift + X
Android layer: Caps Shift + Symbol Shift + C

QWERTY Key List

ZX Spectrum	*Key by itself*	*Caps Shift + key*	*Symbol Shift + key*	*Caps Shift + Symbol Shift + key*
1	1	Escape	!	F1
2	2	Caps Lock	@	F2
3	3		#	F3
4	4		$	F4
5	5	Cursor Left	%	F5
6	6	Cursor Down	&	F6
7	7	Cursor Up	'	F7
8	8	Cursor Right	(	F8
9	9		)	F9
0	0	Backspace	_	Delete
Q	q	Q	<=	F10
W	w	W	<>	F11
E	e	E	>=	End
R	r	R	<	F12
T	t	T	>	Tab
Y	y	Y	[	
U	u	U	]	Page Up
I	i	I		Insert
O	o	O	;	Alt
P	p	P	"	Print Screen
A	a	A	~	Apple/Windows
S	s	S	|	
D	d	D	\	Page Down
F	f	F	{	
G	g	G	}	Alt Gr
H	h	H	^	Home
J	j	J	-	
K	k	K	+	
L	l	L	"="	
ENTER	Return	Return	Return	Return
CAPS	***	***	***	***
Z	z	Z	:	Mac Layer
X	x	X	£	Windows Layer
C	c	C	?	Ctrl
V	v	V	/	Android Layer
B	b	B	*	
N	n	N	,	
M	m	M	.	
SYMBOL	***	***	***	***
SPACE	Space	Break	Space	Space