C64 Keyboard Controlled Kernal Switch Rev. 1

Module Description

Introduction

The **C64 Keyboard Controlled Kernal Switch** is a module that passively watches the scan activity of the C64 keyboard and is capable of setting the higher address bits for Kernal adaptors, similar to project number 123 C64 Kernal-Adaptor/Switch for *Short Boards* that can address up to 7 Kernals, project number 124 C64 Kernal-Adaptor/Switch for *Long Boards* that can address up to 8 Kernals and/or the Character Set Adaptor project number 126 C64 CHARSET-Adaptor/Switch that can address 16 character sets.

A $\overline{\text{RESET}}$ signal is provided to reset the C64 while/after selecting the Kernal via the three most significant EPROM address bits. Two excess IO-pins of the processor Atmel ATmega328p are routed to the pin header J4 for future use.

The ATmega328p can be configured with an Arduino Uno. A pin header provides access to the serial interface for uploading the compiled Arduino sketches. Initial programming is performed via the ICSP connector J2.

The board fits between the keyboard connector of the C64 mainboard and the keyboard itself. All keyboard signals are interconnected and some keyboard signals are monitored by the micro controller.

Watch a brief overview video: http://bit.ly/C64KbKernalSwitch-Intro

Usage of the Kernal Switch

To make use of the Kernal Switch, more than one Kernal should be programmed to the EPROM on the EPROM adaptor. The Kernal can be selected by pressing RESTORE and a number key simultaneously. The Kernal, determined by the number will be selected and the C64 will be reset. This selection is non-volatile and will be effective even after switching off the C64.

Holding down the RESTORE key for a couple of seconds without pressing a number, a reset of the C64 is performed.

In case an empty Kernal memory slot is selected, the C64 will crash, since there is no Kernal found. In this case, the C64 does not perform a keyboard scan and since the Kernal Switch is only listening to the keyboard scanning, but does not perform any keyboard scan, the Kernal number cannot be set manually anymore. After several seconds without keyboard scan, the Kernal switch will automatically recover to the first Kernal and reset the C64. This will recover the system.

Some cartridges will not perform a keyboard scan (such as the Dead Test cartridge). The recovery mechanism will conflict with this kind of cartridge and reset the C64 after several seconds. To circumvent this problem, it is possible to deactivate the Kernal switch by pressing RESTORE **0** (zero). This will switch to the first Kernal and set a non-volatile flag. After switching off the C64 and inserting the cartridge, no keyboard activity timeout will be performed, the cartridge will work normally.

After the cartridge was removed, the first Kernal will be executed and (of course) a normal keyboard scan will be performed. Selecting a Kernal with RESTORE and a number will activate the Kernal Switch again and the new selection will be effective.

Unstoppable RESET

Some games manipulate the RESET vector and the NMI vector of the C64 by writing a CBM80 signature into RAM ($8000). This will prevent a proper reset of the C64 by a reset button or this Kernal switch. To circumvent this problem, the $\overline{\text{EXROM}}$ signal can be asserted while reset and released a short time after reset. This is solving most of the problems of this kind of manipulative software.

Assembly

It is assumed, that the person who assembles the Kernal Switch is experienced in soldering. First the SMD parts should be soldered, the bottom entry receptacle J1 should be soldered last can be aligned to the holes in the board using the pin header J5. Pin2 of J2 has to be clipped (refer to Figure 1) and the connector has to be inserted into the holes from the solder side. Then J1 has to be placed over those pins and the connector has to be aligned in a way that it is centered on the J1 silk screen drawing and the SMD pins are in the middle of the solder pads (refer to Figure 2). Then one pin has to be soldered. Then the connector might need to be corrected and a second pin can be soldered. If the pin header inserts easily from the bottom, all other pins can be soldered.

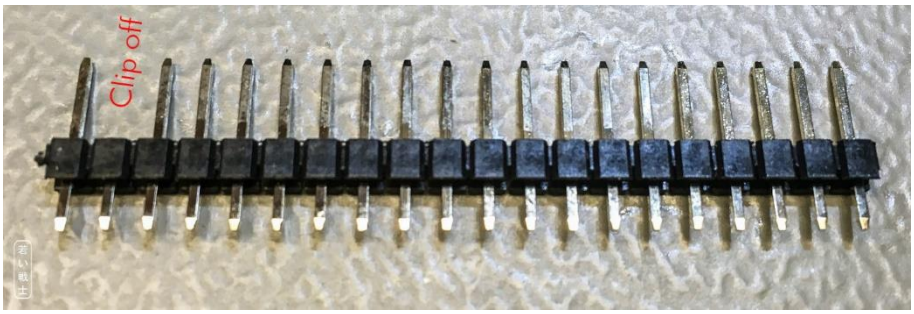

Figure 1: Preparation of J5 (clipping off the 2nd pin)

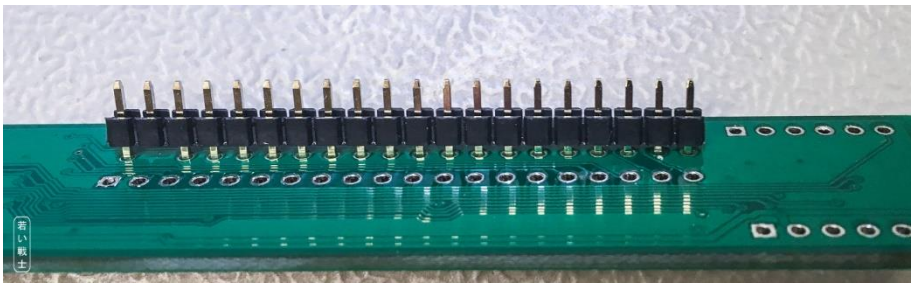

Figure 2: Centering of J1

Finally, the through-hole parts have to be soldered.

Close (or leave open) the solder bridge CP1, depending whether a short board (ASSY 250469, close CP1) or a long board (all other ASSY numbers, leave CP1 open) is present in the C64.

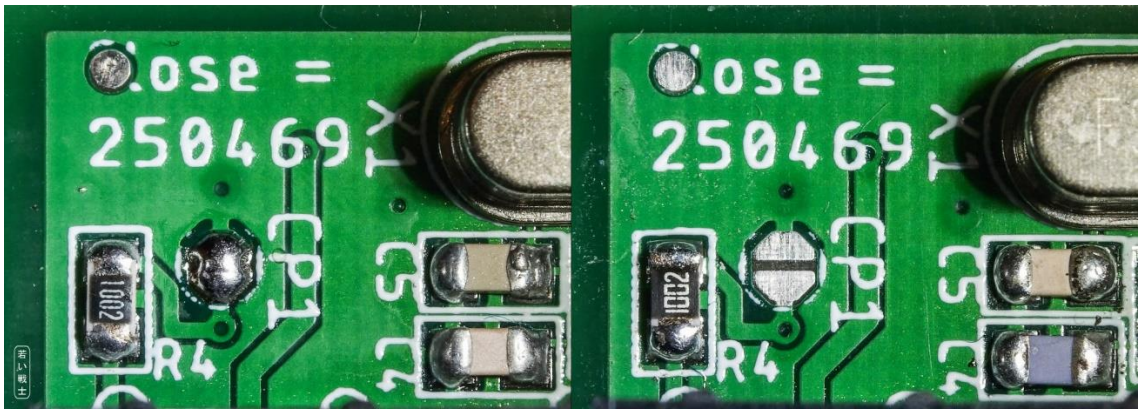

Figure 3: CP1 configured for short board (left) and long board (right)

To prevent the components of the Kernal Switch from making contact with any parts of the C64 mainboards, the area top and bottom of the keyboard connectors can be taped with duct or gaffer tape.

After being completely assembled, the micro controller IC1 (ATmega238P) has to be programmed. Refer to document number 128-6-05-00 (Programming Guide).

For wiring the Kernal Adaptor, the Kernal Switch and the C64, refer to document number 128-6-04-00 (Wiring).

The Assembly of the Kernal Adaptor might require de-soldering the Kernal ROM U4 and installing a socket.

Troubleshooting

In case the C64 does not start properly, make sure that the Kernal Adaptor is inserted properly into the socket and the EPROM is properly programmed. At least the first Kernal has to be programmed. The Kernal Switch should sit firmly on the keyboard connector and do not make electrical contact with any component leads of the C64 mainboard.

The Atmega328P has to be programmed with either the production file or the *.hex and the fuses have to be set properly (refer to document 128-6-05-00: Programming Guide).

Also, it is important to open or close the configuration solder bridge CP1 on the module to determine between a *short* and a *long board*. For C64 *short boards*, the solder bridge has to be closed, the BASIC has to be in the first 8k of the EPROM and a Kernal (original recommended) has to be in the 2nd 8k of the EPROM.

In case the Kernals do not switch in the sequence as they were programmed into the EPROM, most likely, the wiring between the Kernal Switch and the Kernal Adapter is wrong (refer to document 128-6-04-00: Wiring).

Check, if the RESET pin is really connected to RESET of the C64. Again, check the cable to the RESET pin.

In case it is not working at all, remove the cable between the Kernal Switch and the Kernal Adaptor and set jumpers. Set all three jumpers (A13, A14, A15) for a long board and the jumpers A14 and A15 for a short board. In case it is working now, the Kernal Switch is selecting the wrong Kernal (maybe an empty 8k slot). Again, check the cable, measuring the pins A13, A14 and A15 on the J4 with a multimeter. Keep in mind that the first Kernal for a short board

is 001_{BIN} and for a long board it is 000_{BIN}. A 0 here is represented by a low voltage (<0.7V) and a 1 is represented by a HIGH voltage (>4V).

If the problem is not solved, checking the solder joints is required. To see, if the software is running on the micro controller, a Serial Cable and a USB/Serial Adaptor plus the Arduino software (Tools → Serial Monitor, set to 9600 baud) might help.

After power up, the software should send messages via the serial cable. The RESTORE key has to be detected and reported (after holding it down) and a number key will be reported too.

Make sure, that the C64 power supply and the PC are connected and that the cables do not for a big loop, as this might prevent proper function. The serial connection is only for programming (in some cases) and for debugging, so it is not a problem if a ground loop causes problems. It was experienced during testing, that with a big ground loop (a couple of square meters) the C64 did not always start properly.

Keyboard Monitoring

The columns 1-4 and 7 as well as the rows 0 and 3 are connected to IO-pins of the micro controller. The C64 sends a scan bit to the columns, which are HIGH on default. The scan bit is LOW and it is shifted through the columns, one at a time. The rows are read back by the C64. A LOW signal on one or more row bits indicates a pressed key.

The following keyboard scan information can be obtained by the microcontroller:

	Column				
	7	4	3	2	1
Row 0	"1"	"9"	"7"	"5"	"3"
Row 3	"2"	"0"	"8"	"6"	"4"

It is obvious that only the numerals are being monitored. Further the RESTORE key is connected to the microcontroller.

Programming Model

Pin	GPIO	Arduino	Function	Configuration	
30	PD0	D0	RxD	Input	Serial Receive
31	PD1	D1	TxD	Output	Serial Transmit
32	PD2	D2	Row0	Input, weak pull-up	Active LOW, INT0
1	PD3	D3	Row3	Input, weak pull-up	Active LOW, INT1
2	PD4	D4	KSW_A13	Output	Kernal Switch A13
9	PD5	D5	KSW_A14	Output	Kernal Switch A14
10	PD6	D6	KSW_A15	Output	Kernal Switch A15
11	PD7	D7	C64RES	Output	Active HIGH
23	PC0	A0	COL1	Input, weak pull-up	Keyboard column 1
24	PC1	A1	COL2	Input, weak pull-up	Keyboard column 2
25	PC2	A2	COL3	Input, weak pull-up	Keyboard column 3
26	PC3	A3	COL4	Input, weak pull-up	Keyboard column 4
27	PC4	A4	COL7	Input, weak pull-up	Keyboard column 7

Pin	GPIO	Arduino	Function	Configuration	
28	PC5	A5	$\overline{\text{RESTORE}}$	Input, weak pull-up	RESTORE-Key, active LOW
12	PB0	D8	$\overline{\text{SHORT\_BRD}}$	Input	Short Board, active LOW
13	PB1	D9	$\overline{\text{EXROM}}$	output	C64 $\overline{\text{EXROM}}$
14	PB2	D10	RS2	TBD	Reserve IO Pin

Connectors

J1 - Keyboard connector to mainboard

Bottom Entry receptacle (1x20p, pitch 2.54mm) – Harwin M20-7862042

Pin	Signal	Pin	Signal
1	GND	11	ROW1 (PB1)
2	No pin	12	ROW0 (PB0)
3	$\overline{\text{RESTORE}}$	13	COL0 (PA0)
4	+5V	14	COL6 (PA6)
5	ROW3 (PB3)	15	COL5 (PA5)
6	ROW6 (PB6)	16	COL4 (PA4)
7	ROW5 (PB5)	17	COL3 (PA3)
8	ROW4 (PB4)	18	COL2 (PA2)
9	ROW7 (PB7)	19	COL1 (PA1)
10	ROW2 (PB2)	20	COL7 (PA7)

J2 – In System Programming

2x3 box pin header (2.54mm pitch). This connector is used for the initial in-system programming of the ATmega238p (IC1). 89

Pin	Signal	Pin	Signal
1	MISO	2	+5V
3	SCK	4	MOSI
5	$\overline{\text{RESET}}$ (IC1)	6	GND

J3 – Serial/Debug

1x5 pin header (2.54mm pitch). In Arduino mode (after programming the Arduino boot loader), a USB/serial converter can be used to transfer the compiled Arduino sketches and for debugging with the serial monitor.

Pin	Signal	USB/Serial converter
1	GND	GND
2	+5V	+5V
3	TxD (output)	RX
4	RxD (input)	TX
5	DTR (input)	DTR

J4 – IO-Connector

1x6 pin header (2.54mm pitch)

This connector outputs the control signals of this board. It is for being connected to the Kernal and/or keyboard adapter. It also provides two reserved IO-Pins for later use.

Pin	Signal	Function
1	KSW_A13	Kernal Switch/Adapter A13
2	KSW_A14	Kernal Switch/Adapter A14
3	KSW_A15	Kernal Switch/Adapter A15
4	$\overline{\text{C64\_RESET}}$	RESET signal for the C64 (to be connected to a RESET point on the mainboard)
5	$\overline{\text{EXROM}}$	$\overline{\text{EXROM}}$ signal for the C64 (to be connected to an EXROM point on the mainboard)
6	RESIO	Reserved IO
7	GND	GND pin for multi-purpose use

J5 - Keyboard connector

Pin header (1x20p, pitch 2.54mm) – Pin 2 clipped off (no pin)

Pin	Signal	Pin	Signal
1	GND	11	ROW1
2	No pin	12	ROW0
3	$\overline{\text{RESTORE}}$	13	COL0
4	+5V	14	COL6
5	ROW3	15	COL5
6	ROW6	16	COL4
7	ROW5	17	COL3
8	ROW4	18	COL2
9	ROW7	19	COL1
10	ROW2	20	COL7

Documents

Document	Document Number
Disclaimer	-
Module Description (this document)	128-6-01-**
Schematic	128-1-01-**
PCB prints	128-2-01-**
Functional Description	128-6-02-**
Software Description	128-6-03-**
Wiring	128-6-04-**
Programming Guide	128-6-05-**
Testing	128-6-06-**
Bill of Material	128-5-01-**.*

** depends on revision

Revision History

Rev. 0

- Prototype (fully functional)

Rev. 0 → Rev. 1

- J4, pin 5 is now for $\overline{\text{EXROM}}$
- J4 has an additional GND Pin (useful in case RESIO is connected to an LED)
- Series resistor in $\overline{\text{EXROM}}$ and RESIO signal (220Ω or whatever is required).
- Additional 1KΩ (R10) between GND and +5V for faster discharge