MG008 ZX Spectrum Adapter for RC2014

What is it?

MG008 is an adapter to enable ZX Spectrum peripherals to be connected to an RC2014. It is designed to overcome many of the problems that arise when trying to get a ZX Spectrum peripheral to work with RC2014:

1. Both the standard RC2014 Serial I/O and most ZX Spectrum peripherals have incompletely decoded address lines[1]. This means there is a significant likelihood that a peripheral will clash with the RC2014 Serial I/O
2. Some peripherals send out a "ROMCS" signal, and then expect to "own" the lower 8k of address space. With a standard RC2014, this will clash with the ROM
3. The ZX Spectrum uses all 16 address bits for I/O for some peripheral devices. RC2014 (or at least, the standard BASIC) only uses the traditional Z80 lower 8 bits
4. The ZX Spectrum 5V output to peripherals is quite weedy, and many peripherals use the 9V output instead (usually converting it down to 5V using a linear regulator). The RC2014 bus does not feature a 9V line[2]

How it does this is as follows:

1. There are numerous fully decoded RC2014 serial I/Os out there (including the MG010) which will solve the first half of this problem. For the second half, MG008 is able to bring in a SERACT (Serial Active) signal from MG010 and use it to gate /IOREQ out to the peripheral[3]
2. MG008 has a LED to show when ROMCS is active, and a jumper to enable the first and second 16k of address space to the peripheral to be transposed (in other words, the peripheral will appear in the empty address space between ROM and RAM on a standard Classic RC2014)

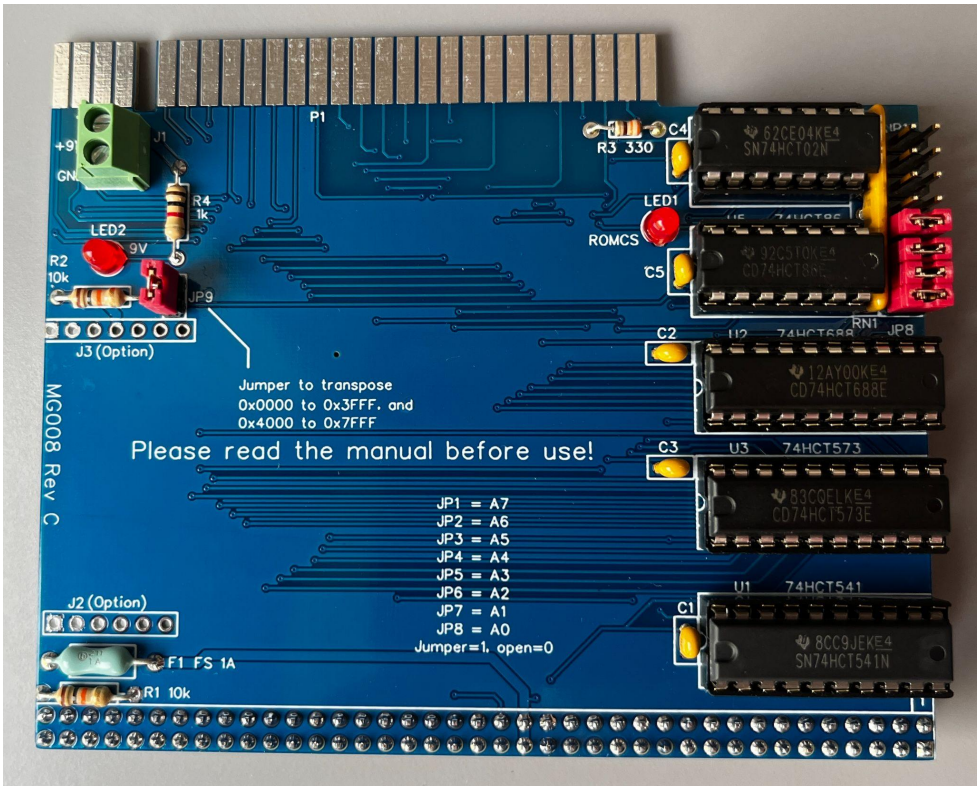
[1] It's not unusual for a ZX Spectrum peripheral to be connected to only 3 or 4 of the 16 address lines

[2] For the sake of completeness, the ZX Spectrum also runs at a much slower CPU clock speed than the Classic RC2014. I have not found this to be an issue with any of the peripherals I have tested, but if it does cause issues it's easy enough to clock the RC2014 slower

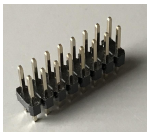
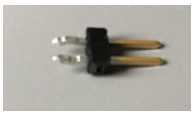
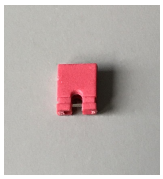
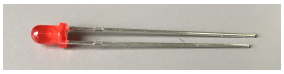
[3] Although I have found one peripheral that does not connect to /IOREQ. Fortunately its addressing was fine without needing gated /IOREQ

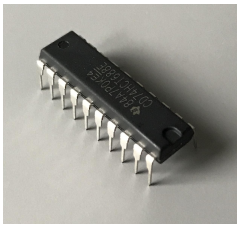
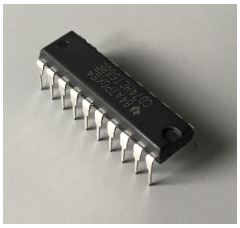
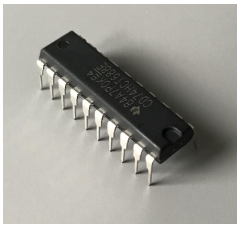
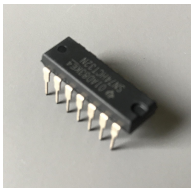
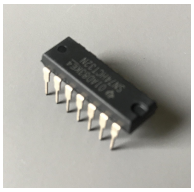
3. MG008 provides an addressable latch, which can be used to apply the upper 8 bits to the address bus during I/O operations
4. MG008 provides a 9V input terminal block to power peripherals where required

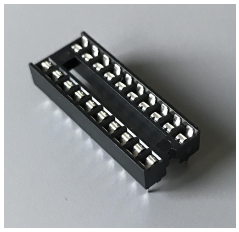
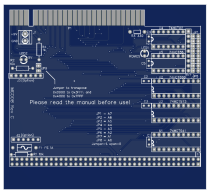
It is best to think of MG008 as a tool to enable experimentation with ZX Spectrum peripherals, rather than a device that is guaranteed to get all modes of any peripheral working with RC2014



What's in the kit?

Name	Quantity	Description	Picture	Present?
C1-5	5	Capacitor, ceramic, 100 nF		❜
F1	1	Fuse, PCB Leaded, 1 A, 125 V		❜
J1	1	Terminal, screw		❜
JP1-8	1	Header, male, 2 x 8 pin, straight		❜
JP9	1	Header, male, 1 x 2 pin, straight		❜
JP1-9 Shunt	9	Jumper shunt		❜
LED1,2	2	LED, Red, Through Hole, T-1 (3mm), 20 mA		❜
R1,2	2	10k Resistor, 250mW		❜
R2	1	330R Resistor, 125mW		❜
R4	1	1k Resistor, 250mW		❜
RN1	1	Fixed Network Resistor, 10 kohm		❜

U1	1	74HCT541		,
U2	1	74HCT688		,
U3	1	74HCT573		,
U4	1	74HCT02		,
U7	1	74HCT86		,
U4,5 sockets	2	14-pin DIP socket		,

U1-3 sockets	3	20-pin DIP socket		,
P2	1	Header, male, 2 x 40 pin, straight		,
PCB	1	MG008 PCB		,

How do I build it?

There's a good chance you will have some soldering experience, as you're likely to have built an RC2014 or equivalent to plug your MG011 into. If you haven't, I recommend searching for an online tutorial, there are some good ones on YouTube.

Recommended tools include:

- Soldering iron (ideally temperature controlled)
- Multicore solder
- Small snips to cut off leads
- Small pliers
- Desoldering pump and/or braid
- Anti-static wrist strap (or steer clear of materials that cause static and touch a grounded object every now and then).

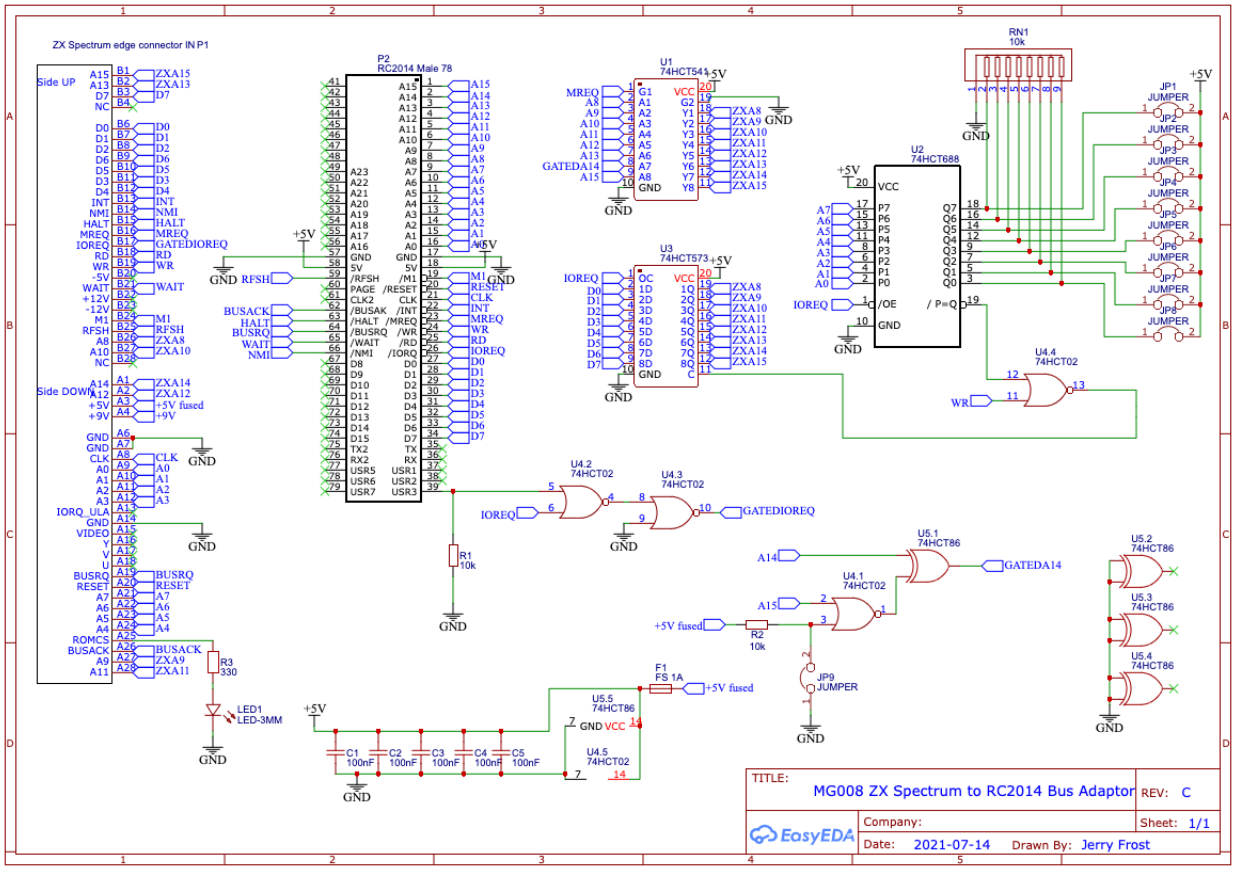
The normal rule of thumb is to solder the lowest height components first, working up:

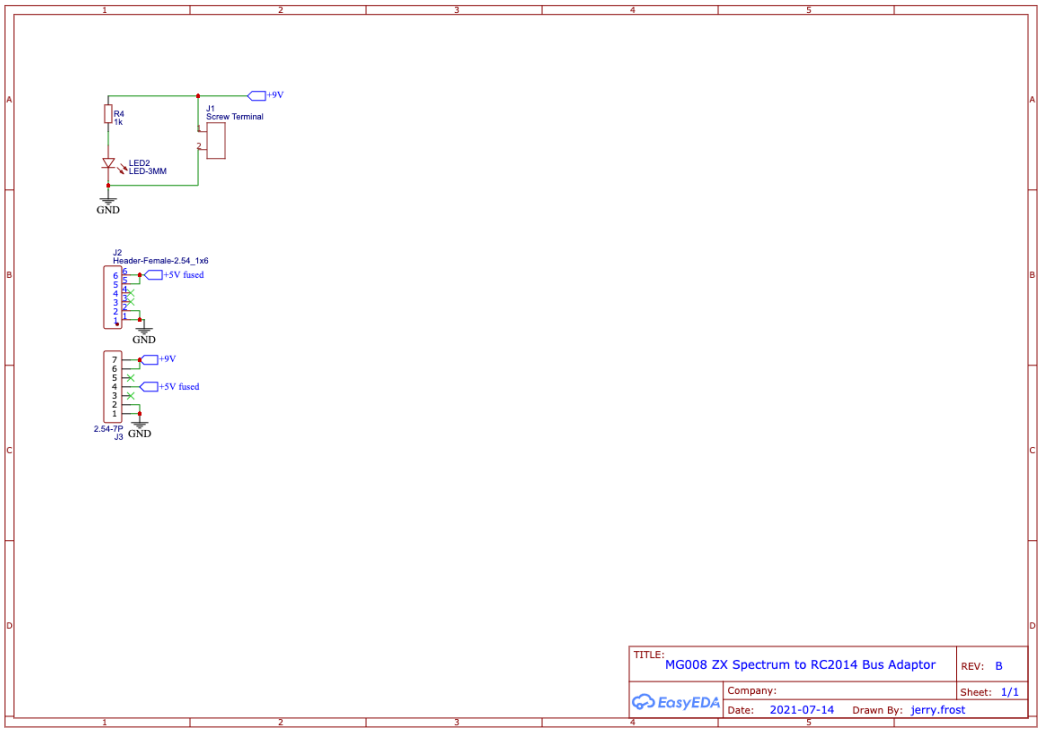
- R1-3. Orientation doesn't matter.
- F1. Orientation doesn't matter.
- P2. This is normally supplied with 80 pins, therefore two need to be carefully cut off using a sharp knife. **Please note P2 needs to be soldered on the other side of the PCB to other components.** Solder one joint only, check the alignment, melt solder and correct alignment if required before soldering remaining joints.
- C1-5. Orientation doesn't matter
- Sockets for U1-5 (do not fit ICs yet). Similarly to P2, solder two opposite corners, check the socket is flat on the board before continuing. Make sure the notches at the end of the sockets match with the PCB graphics, to reduce the risk of installing the ICs the wrong way round
- RN1. Note that the dot on one end should align with the marking on the PCB RN1 graphic
- LED1-2. Make sure the flat side is aligned with the graphic on the PCB
- JP1-8 (actually a single part), JP9
- J1

If you have flux cleaner, clean all joints. Now inspect them carefully for issues (a magnifying glass of some sort can be very helpful, the camera on some phones works quite well).

The final step prior to plugging into the host system and testing is to fit the ICs into their sockets. Their legs will probably need a bit of gentle bending on a table or similar surface, to bring the two rows a little closer to each other. Pay attention to orientation (even after all this hard work, it’s easy to get wrong).

How does it work?





RC2014 and the ZX Spectrum are both Z80-based, therefore the majority of connections pass straight through. The exceptions are:

- If MG010 is present in the system, U4.2 and U4.2 take the SERACT signal from USR3 (which is the MG010 saying it's busy doing I/O) and use it to disconnect IOREQ from the peripheral
- LED1 shows that ROMCS is active, meaning that the peripheral wants to use the lower 8k of address space
- When JP1 is jumpered and A15 is at logic zero, U4.1 and U5.1 invert the A14 signal going out to the peripheral. This moves the peripheral from the lower 16k of address space to the next 16k block up, enabling it to be peeked and poked
- U1 passes the upper 8 bits of address bus out to the peripheral unchanged when a memory request is active, but disconnects this during I/O requests
- When I/O requests are active, the 8 bits loaded into U3 are connected to the upper 8 bits instead.
- U2 and JP1-8 set the address of U3, via which the value above can be loaded
- JP1 allows 9V to be fed to the peripheral from an external power source. J2 and J3 are designed to enable connection of a 5V to 9V switched mode power supply PCB, to make MG008 self contained. I have prototyped and tested this, but it would cost as much as MG008 so I haven't completed the design. If one or two people contact me and say they would like this capability, I would definitely consider finishing it

How do I use it?

Before plugging anything in, research the peripheral:

- What power does it need?
- What address(es) does it use?
- Is it accessed by I/O or memory requests?
- What commands (if any) does it expect?

This should not be too difficult, there are a wealth of websites out there giving all these kinds of details. Worst case (if you're happy to do this), opening up the peripheral and examining which pins are connected will reveal a lot of information

An address needs to be set to allow U3 to be loaded. To use an example:

Jumper	A7	A6	A5	A4	A3	A2	A1	A0
Status	Open	Open	Open	Open	Jumper	Jumper	Jumper	Jumper
Value	0	0	0	0	8	4	2	0

Summing the values gives a decimal address of 15. OUT 15, X will load an 8 bit number X ready for any I/O requests to the peripheral.

Take care plugging in

When connecting to a RC2014 with single row connectors, then the row of MG008 P2 contacts closest to the edge of the PCB needs to be plugged in. There are signals coming out of the ZX Spectrum edge connector that are present in the other row of P2, and they have all been brought through. Having said that, I have yet to find a peripheral that uses them.

On the other end, I have tried to make the keying notch on P1 the right size, but there's no specification for this dimension (that I can find), and the key in the connector varies hugely from peripheral to peripheral. Therefore be a little careful with alignment as you plug the peripheral in. These edge connectors were not particularly reliable when new, and they haven't got any better with age. It's worth checking for bent contacts and

unplugging/refitting the peripheral a few times to get the best connection possible.

Think about power

If the peripheral only needs 5V, then you should have no worries. A typical RC2014 setup should be able to supply more 5V current than a ZX Spectrum ever could.

If the peripheral needs 9V, then a 9V power supply will be needed that is happy being connected low side to low side with whatever 5V supply is being used for the RC2014. If RC2014 is being powered by 9V (using an onboard regulator to supply RC2014 5V), then an easy way to proceed is to connect the high side of that 9V (and ideally the low side) to screw terminal J1. I have seen peripherals take in excess of ½ amp, which must have been pushing the standard Spectrum 9V power supply brick quite a bit.

Example 1 - Kempston Joystick Interface

This is extremely easy, as it responds to standard INP(31) commands:

BASIC CODE

```
10 LET A = INP(31)
20 PRINT A
30 GOTO 10
```

Values produced are:

1	Right (R)
2	Left (L)
4	Down (D)
8	Up (U)
16	Fire (F)

Example 2 - Cursor Mode Joystick Interface

This interface simulates the cursor keys on the keyboard. These need to be read using 16 bit I/O addresses:

EFFEh (61438 decimal): Down, Up, Right, Fire
F7FEh (63486 decimal): Left

EF00h is 61184, which means 239 (61184/256) needs to be loaded into U3 for the upper 8 address bits, and "INP(254)" will then read FE for the lower 8 bits.

F700h is 63232, which means 247 (63232/256) needs to be loaded into U3 for the upper 8 address bits, and "INP(254)" will then read FE for the lower 8 bits.

(Examples assume MG008 address is set to 15 decimal)

BASIC CODE	*Notes*
10 OUT 15,239 20 LET A = INP(254) 30 PRINT A	*Set upper 8 address bits to "EFXX"* *Read EFFE*

BASIC CODE	*Notes*
10 OUT 15,247 20 LET A = INP(254) 30 PRINT A	*Set upper 8 address bits to "F7XX"* *Read F7FE*

Values produced for EFFE are:

U = 23 (10111)
D = 15 (01111)

R = 27 (11011)
F = 30 (11110)

Values produced for F7FE are:

L = 15 (01111)

Example 3 - Currah Microspeech

Please note this device needs 9V power.

This is a device that sends out a "ROMCS" signal, and then expects to "own" the lower 8k of address space.

Normal operation is that it is toggled on or off by accessing 0038h (read or write). The device can then be accessed to check status or send allophones at 1000h. I put a short intro to allophones and their usage in my instructions for MG005 (on my website).

I use this device by starting off with JP9 jumpered, to move it (in address terms) away from the RC2014 ROM. Jumpering JP9 adds 4000h to any read/write to the Microspeech:

- On/off is now 0038h + 4000h = 4038h or 16440 decimal
- Device access is not 1000h + 4000h = 5000h or 20480 decimal

Before running the below code, the Microspeech must be toggled on (if it did not start in that state) by sending "PEEK(16440)" so that the ROMCS LED lights.

BASIC CODE	*Notes*
10 DATA 27,7,45,15,53,3,46,51,45,1,21,3	*Allophones for "Hello World"*
20 LET LE=12	
30 DIM XX (LE)	*Make an array XX to hold them*
40 FOR Y=1 TO LE	
50 READ XX(Y)	*Fill array with allophones*
60 NEXT Y	
70 FOR Z=1 TO LE	*Loop through allphones*
80 IF (PEEK(20480) AND 1)=0 THEN GOTO 100	*Check whether device is busy, if not GOTO 100*
90 GOTO 80	
100 POKE 20480,XX(Z)	*Send allophone to device*
110 NEXT Z	*Go round for next one*

Acknowledgements/Legal

MG008 has been designed for RC2014 with reference to the RC2014 Module Template. All pinouts used are in compliance with the RC2014 Module Template.

RC2014 is a trademark of RFC2795 Ltd.

MG008 has been designed for hobbyist use only and is not to be used for safety or business critical applications.

MG008 is designed for experimentation and is not guaranteed to get any given peripheral, or any given mode of operation of a peripheral working. ZX Spectrum software will of course not run on RC2014, so any functions enabled by specific software routines will not work out of the box.

Plugging 1980's peripherals into a RC2014 system is not risk-free, especially if they are not known to be working. It is possible the RC2014 may be damaged. Having said that, you may wish to plug an unknown-condition peripheral into your RC2014 first, rather than your priceless Issue 0 ZX Spectrum….