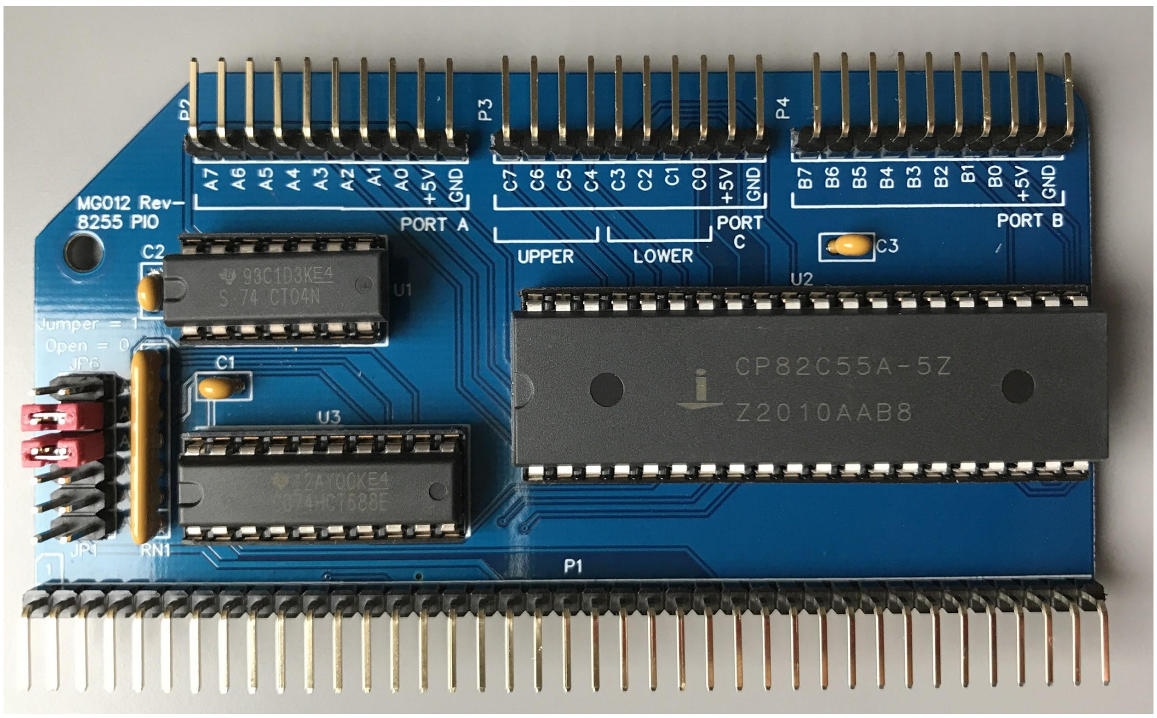
MG012 Programmable Input/Output

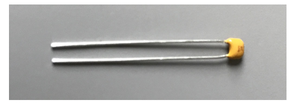
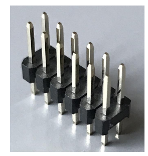
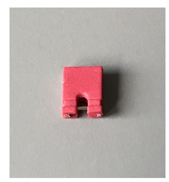
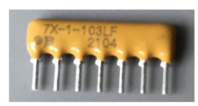
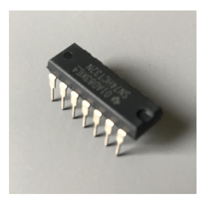
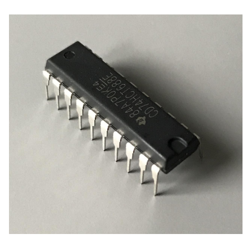
What is it?

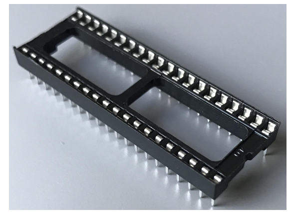
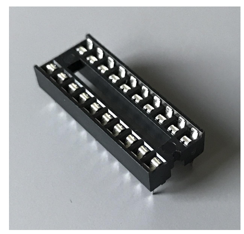
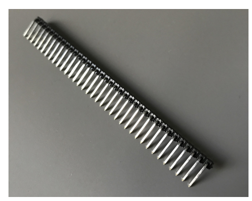
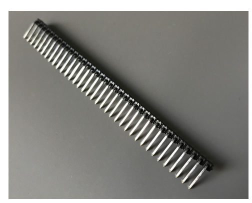
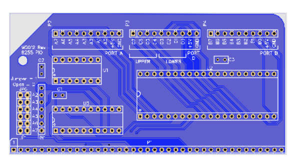
MG012 is a Programmable Input/Output (PIO) designed for RC2014. It uses the 82C55 Programmable Peripheral Interface to provide 24 I/O pins, which can be flexibly configured as input or output.

MG012 can be addressed using standard BASIC input and output commands, with one address being used for a control word, and three more for three groups (A, B and C) of eight I/O pins.



What’s in the kit?

Name	Quantity	Description	Picture	Present?
C1-3	3	Capacitor, ceramic, 100 nF		•
JP1-6	1	Header, male, 2 x 6 pin, straight		•
JP1-6 Shunt	6	Jumper shunt		•
RN1	1	Fixed Network Resistor, 10 kohm		•
U1	1	74HCT04		•
U2	1	82C55A (option for buyer to supply)		•
U3	1	74HCT688		•

U1 socket	1	14-pin DIP socket		
U2 Socket	1	40-pin DIP socket		
U3 socket	1	20-pin DIP socket		
P1	1	Pin Header, Right Angle		
P2-4	1	Pin Header, Right Angle (See “how do I build it”)		
PCB	1	MG012 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 MG012 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).

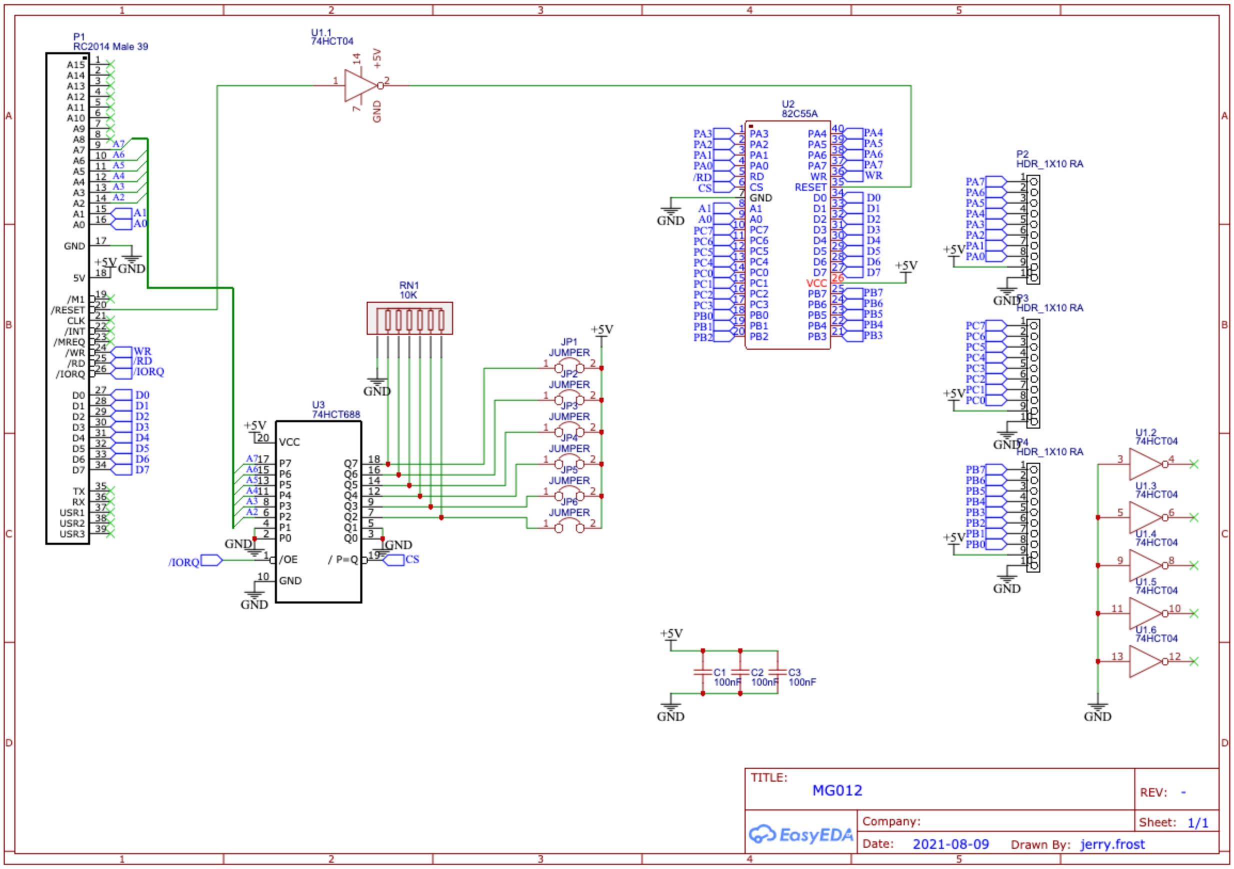
The normal rule of thumb is to solder the lowest height components first, working up:

- P1. This is normally supplied with 40 pins, therefore one needs to be carefully cut off using a sharp knife. Then, solder one joint only, check the alignment, melt solder and correct alignment if required before soldering remaining joints. If supplied with 36 pins, fit at the Pin 1 end of the PCB
- P2-4. These are supplied as a single 40 pin assembly (the same as P1). Three 10-pin assemblies need to be cut from this using a sharp knife or snips
- C1-3. Orientation doesn't matter
- RN1. Note that the dot on one end should align with the marking on the PCB RN1 graphic
- Sockets for U1-3 (do not fit ICs yet). Similarly to P1, 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
- JP1-6 (actually a single part)

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).

<u>How does it work?</u>



JP1-6 are used to set the address of MG012. When the address bus (upper 6 bits) matches this and /IORQ is active, U2 outputs a "/P=Q" signal. This enables U2 for a read or write over the 8 bit data bus, depending on whether /RD or /WR are active.

U1.1 inverts the reset signal (U2 needs an active high) and sets U2 to "all ports input" after a reset.

U2 does all the heavy lifting, it has 24 I/O pins which may be individually programmed in 2 groups of 12 (A plus C Upper, B plus C Lower) and used in 3 major modes of operation. It's not possible to do justice to all the possibilities here, but they are covered in depth in the Renesas datasheet (and also several online tutorials):

https://www.renesas.com/us/en/document/dst/82c55a-datasheet

The following section ("How do I use it") will give the basics of getting started in Mode 0.

<u>How do I use it?</u>

The first step is to set the address. If you are using the standard RC2014 serial I/O board, then setting A7 to zero (jumper removed) ensures the MG012 will not clash with it. If you have other I/O boards fitted, then you will need to take them into account also.

To use an example:

Jumper	A7	A6	A5	A4	A3	A2
Status	Open	Open	Open	Jumper	Jumper	Open
Value	0	0	0	16	8	0

Summing the values gives a decimal address of 24 (A0 and A1 both zero) to 27 (A0 and A1 both 1). These are used as follows:

- 24 - Port A
- 25 - Port B
- 26 - Port C
- 27 - Control Word

The basics of the Control Word in Mode 0 are:

Bit	7	6	5	4	3	2	1	0	Dec.
Port (1=IP, 0=OP)				A	C Upper		B	C Low	
Value (all ports IP)	1	0	0	1	1	0	1	1	155
Value (A=OP, B,C=IP)	1	0	0	0	1	0	1	1	139

Reading address 27 after a reset will get a result of 155 (all ports IP). Using the above it should be easy to set the different ports as input or output as required. For example, writing 139 to address 27 will enable Port A as an output, after which the binary values of outputs to address 24 will appear at Port A pins. Similarly, reading addresses 25 and 26 will give the inputs that are being applied to Ports B and C respectively.

<u>Example 1, Loopback Test:</u>

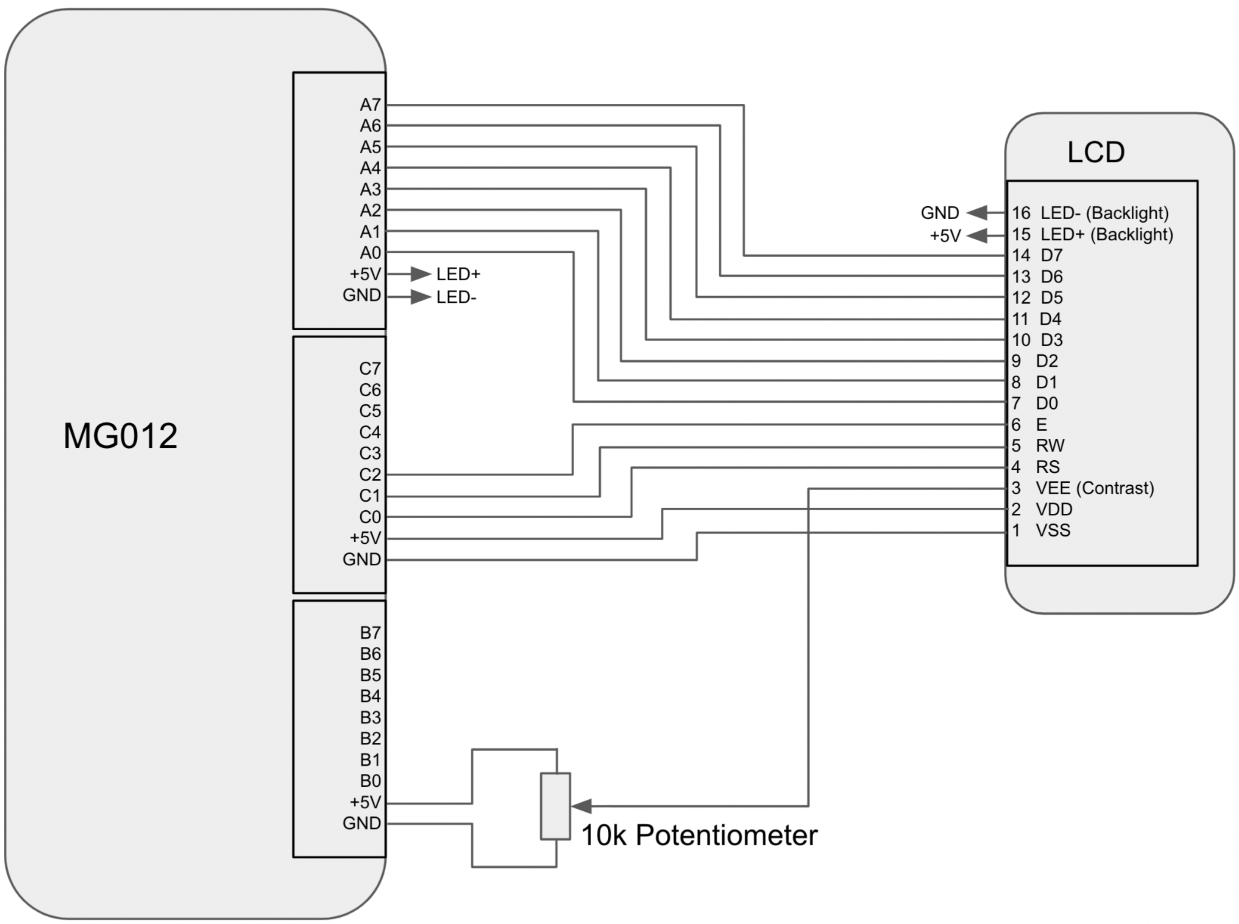
A quick example follows (using the addresses of 24 to 27). It’s a loopback test, and assumes A0 to A7 are connected to B0 to B7 respectively. It uses MG011 at address 30 to provide a random number between 0 and 255, but it could be rewritten to use the BASIC RND function, or to count up by one every time.

BASIC CODE	*<u>Notes</u>*
`10 PRINT "A to B"`	
`20 OUT 27,139`	*Set up Port A as OP, B,C as IP*
`30 FOR A = 1 TO 1000`	*Loop for 1000 numbers*
`40 LET B = INP(30)`	*Grab random number from MG011*
`50 OUT 24,B`	*Output number on Port A*
`60 LET C = INP(25)`	*Input number on Port B*
`70 IF B<>C THEN PRINT "FAIL",B," ",C`	*Compare, and warn if different*
`80 NEXT A`	
`90 PRINT "RUN COMPLETE"`	

Example 2, LCD Display:

This is a more "real world" example (the loopback test is a good example of input and output that doesn't need additional hardware (beyond wires), but it's not the most exciting).

The LCD display in question is a 16 x 2 digit 5V parallel interface LCD module. There are many slightly different variants of this out there, they feature configurable (8 or 4 bit) parallel buses (we will use 8 bit). The hookup uses A and some of B port, plus various 5V and ground connections:



Port C (OUT26, X) is connected as follows:

Port C Pin	Binary Value (X)	Connected to	Usage
C0	1	RS	Register select. 0 writes into instruction register, 1 into data register (for writing in text to be displayed)
C1	2	RW	Read/Write. We will only use 0 = write
C2	4	E	Enable input. A 1 to O transition on this pin writes (or reads) the contents of the data bus to or from either the instruction or data register

BASIC CODE	*Notes*
1000 REM Testprogram 8255 with LCD	
1010 GOSUB 10000:REM Init LCD	*Go to subroutine to initialise LCD*
2000 REM Main-Program	
2010 TX$="82C55 I/O MG012"	*Text to be displayed in upper*
2020 UX$="Type C to clear"	*and lower LCD lines*
2030 GOSUB 10300	*Go to subroutine to write*
2040 INPUT "Type C to clear";C$	
2050 IF C$="C" THEN GOTO 2080	*Typing C goes to subroutine*
2060 IF C$="c" THEN GOTO 2080	*to clear LCD*
2070 GOTO 2040	
2080 GOSUB 10200:REM clear LCD	
2090 STOP	
10000 REM Init LCD	*Subroutine to initialise*
10010 OUT 27,138 :REM 8255 all ports Output	*Sets all 3 ports to output mode*
10020 OUT 24,0	
10040 OUT 26,0	
10050 FOR I=1 TO 4	
10060 OUT 24,56 :REM 8Bit	*Put 56 onto 8 bit bus*

10070 OUT 26,4 10080 OUT 26,0 10090 NEXT I	*Toggles port C bit 4 ("E") with bit 1 at zero (clocks 56 into instruction register to set 8 bit mode)*
10100 OUT 24,6 :REM Increment 10110 OUT 26,4 10120 OUT 26,0	*Similarly, clocks 6 into instruction register (cursor will move to right)*
10130 OUT 24,12 :REM Cursor off, Display on 10140 OUT 26,4 10150 OUT 26,0	*Similarly, clocks 12 for display on, cursor not shown*
10160 GOSUB 10200:REM Clear display 10170 RETURN	*Clear display and return*
10200 REM Clear display 10210 OUT 24,1 10220 OUT 26,4 10230 OUT 26,0 10290 RETURN	*Subroutine to clear display* *Clocks 1 into instruction register to clear display*
10300 REM Output string tx$ to LCD	
10310 L=LEN(TX$)	*Work out length of string*
10320 FOR I=1 TO L	*Loop through it*
10330 T=ASC(MID$(TX$,I,1))	*Let T = next character*
10340 OUT 24,T	*Put T onto 8 bit bus*
10350 OUT 26,5	*Clock T into data register*
10360 OUT 26,0 10370 NEXT 10400 REM Output string ux$ to LCD	
10410 OUT 24,168 10415 OUT 26,4 10417 OUT 26,0 10420 L=LEN(UX$) 10430 FOR I=1 TO L 10440 T=ASC(MID$(UX$,I,1))	*Same process all over again for lower line*

```
10450 OUT 24,T
10460 OUT 26,5
10470 OUT 26,0
10480 NEXT
10490 RETURN
```

Timings and Compatibility

MG012 is supplied with either 5MHz or 8MHz 82C55 devices, depending on what's available (they are still being manufactured, price is essentially the same for either, but availability can be patchy). I have examined the timing diagrams and confirmed that either device should work equally well.

MG012 does not comply fully with 82C55's need for address/Chip select inputs to be stable 20ns after a write (by about 10ns). I have tested extensively and noticed no issues, and the majority of Z80/82C55 designs and implementations on the web have the same "feature", and are also apparently working fine.

More recently, I have tested MG012 (and three different 82C55s) for many hours with a Z80 based RC2014 running at 20MHz, running loopback tests. No failures at all have been logged. I have no means of testing against Z180 processors, but suspect results will be similar.

The only timing caution I would note is where users source their own 82C55 from Chinese or similar suppliers on eBay or elsewhere. The risk of these being fakes is high, sadly. If you decide to go this route, then I obviously can not guarantee the results.

Acknowledgements/Legal

MG012 has been designed for RC2014 with reference to the RC2014 Module Template. All pinouts used and the physical outline are in compliance with the RC2014 Module Template.

The BASIC code for Example 2 is based on code posted by Thomas Riesen in the RC2014-Z80 Google group.

MG012 has been designed for hobbyist use only and is not to be used for safety or business critical applications.