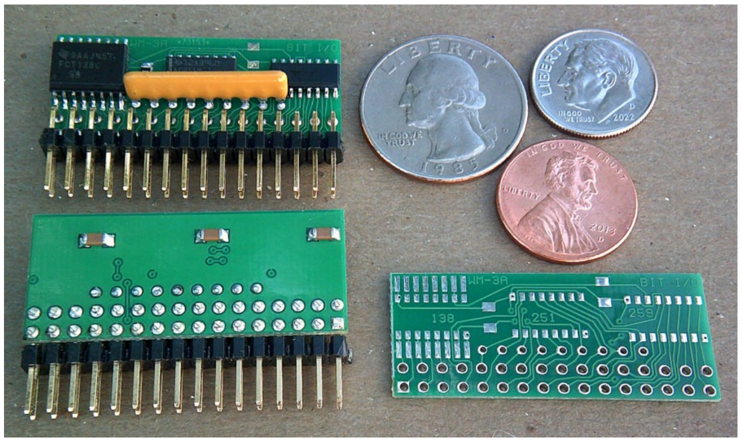
WM-3a 65xx-bus-compatible Bit-I/O Module



- 8 input bits and 8 output bits, intended for where bits are twiddled or tested independently of others in a group

- bus-compatible with 6502, 65816, and other 65-family processors

- Use it like an IC; but it takes only .32 sq. in. of board space, as much as a 16-pin DIP.

- manage individual bits faster than you can with a 65c22 VIA (but it's intended to *augment*, not *replace,* the VIA)

- Set or clear any output bit by its "set" address or its "clear" address, with a single machine-language store instruction, without regard for what data is supposedly being stored, and without affecting the other bits or any processor registers. (No need for R-M-W.)

- test any input bit by its address, with a single `BIT` instruction (affecting the N or V flag, from the lone data bit which is output-only and can be connected to your D7 or D6), neither regarding nor affecting the processor's A, X, or Y, nor having to AND-out irrelevant bits

- four "chip" (module)-select lines, allowing address-decode logic to be simpler and faster

- A logic low on the RST\ input clears the output bits.

- Built with 5V TI CY74FCT138CTSOC (with 5ns max prop. delay), TI CD74AC251M96 ('AC**T**251 not available), and OnSemi MC74ACT259DR2G. (Bare boards also available for you put put your own parts on. The land pattern for the '138 accommodates either the standard or the wide SO-16.)

- intended for homebrew computers

- two ground pins and two Vcc pins, distributed so no signal pin is more than .2" away from a ground or bypassed Vcc pin

- maximized high-speed performance of two-layer board with ground plane, and power-supply bypass capacitors mounted under the Vcc pin of each IC, with power and ground vias in the pads

- hobbyist-friendly 32-pin dual-row header of .025" square posts on .100" centers, to plug into readily available, inexpensive thru-hole sockets on your prototyping perfboards

- 1.600" long x 0.680" wide, not including pin header

- Available with straight or 90° pins. Straight pins will put the module parallel to the motherboard, still allowing parts underneath, while 90° pins will make it stand up perpendicular to the mother board.

- Every complete module is fully functionally tested. I have no way to test for nanosecond timing margins, but you can look up the manufacturers' guaranteed margins in the data sheets of the three ICs, linked in the pages linked above, and the schematic is included below, on page 5.

How it's useful for the intended application:

In the kind of work I do with the workbench computer, there are many cases of wanting to set or clear an output bit, or test an input bit, without regard to other bits in an 8-bit port. Here are some examples:

- To read the fast A/D converter, I set the enable bit true (which is just a single bit), then read the value the converter puts on an 8-bit VIA port, then set the enable bit false again. One of these individual output bits could be used for the enable, without ANDing and ORing it into a VIA port and taking up some of an 8-bit port for it. Multiple A/D's could be used this way, all connected to the same VIA port but using their own enables from these individual bits.

- In the case of a beeper, if I'm not using a VIA's T1 to automatically toggle the line (which I don't normally do anyway), I could put the beeper on one of these individual bits.

- Individual bits could be used for annunciator LEDs, whether for troubleshooting or anything else, or triggering some other event.

- In the case of 65SIB, one bit could be used for config. If you're bit-banging, other bits come in.

- In the case of the parallel printer port, you could use these bits for the strobe, fault, busy, etc..

- In the case of a battery-life tester I made for an intercom, an input bit could be used to monitor the intercom's low-battery LED, not just to see when it started coming on, but also to test the flash rate (in software).

- In the case of our Bluetooth programmer, one of its input lines is a command-mode bit. One these output bits could be used to set that.

- One project involved a jar with a light bulb in it as an oven, to test a jack for days to make sure it wouldn't deform if it were in an airplane parked on the tarmac in Phoenix in the summer, with a plug still in the jack. I had to make sure it would handle more heat that it would ever get. I put an LM335 temperature sensor on the jack. Software turned the light on and off to keep the jack at 100°C. One of these individual output bits could be used to control the relay to turn the light on and off.

- Bit-banging synchronous-serial interfaces like SPI becomes faster.

The possibilities are endless. I did all the above with 65c22 VIAs, before I had this module. Not all of them would benefit from its speed advantage (by way of fewer instructions to do the job), but it would nevertheless leave VIA pins available for other uses.

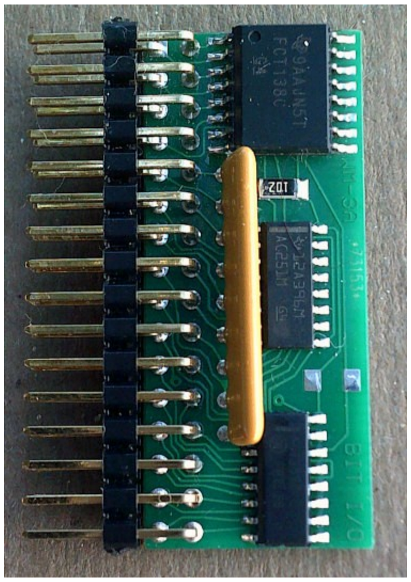
Pinout:

The order may not be ideal, but it's because of the pinouts of the ICs and the layer limitation and size limitation applied to the layout. Since the usage is for individual bits independently of the others, this should not be a problem, as you can connect the socket pins on the motherboard whichever way is convenient, then set your EQUates in the source code accordingly.

The connections in the schematic may look mixed up. However, the input numbers and output numbers on the module's pins are adjusted for this, so you can ignore the strange internal connections.

Holding the module as shown in the picture on the right, component side toward you and with the pin header along the left edge, pin 1 of the pin header is on the top left (near the corner of the board), 2 is to its right and .100" farther from the corner of the board.

```
pin 1   R/W  □ □  CS3  pin 2
        CS1\ □ □  ϕ2
        gnd  □ □  CS2\
        A0   □ □  CS0
        I1   □ □  I6
        I4   □ □  I2
        I0   □ □  +5V
        I7   □ □  D7
        A1   □ □  I5
        +5V  □ □  I3
        O0   □ □  A2
        O7   □ □  RST\
        O2   □ □  A3
        O5   □ □  gnd
        O6   □ □  O1
pin 31  O3   □ □  O4   pin 32
```



Note: This is looking at the component side of the module. If you look into the holes of the socket these pins go into on the motherboard, the rows will appear reversed, ie, that looking into the socket, pin 1 will be on the top-right, not the top-left, pin 2 will be on the top-left, and so on, until you reach pin 32 on the bottom left.

Operation:

For writing to the 74xx259, each output bit gets two addresses, one address to set the bit and the other to clear it. You just write to it. It doesn't see the data bus, only the address, so data is irrelevant. (Actually, A0 <u>is</u> the data input.) Other bits are not affected, because they have different addresses.

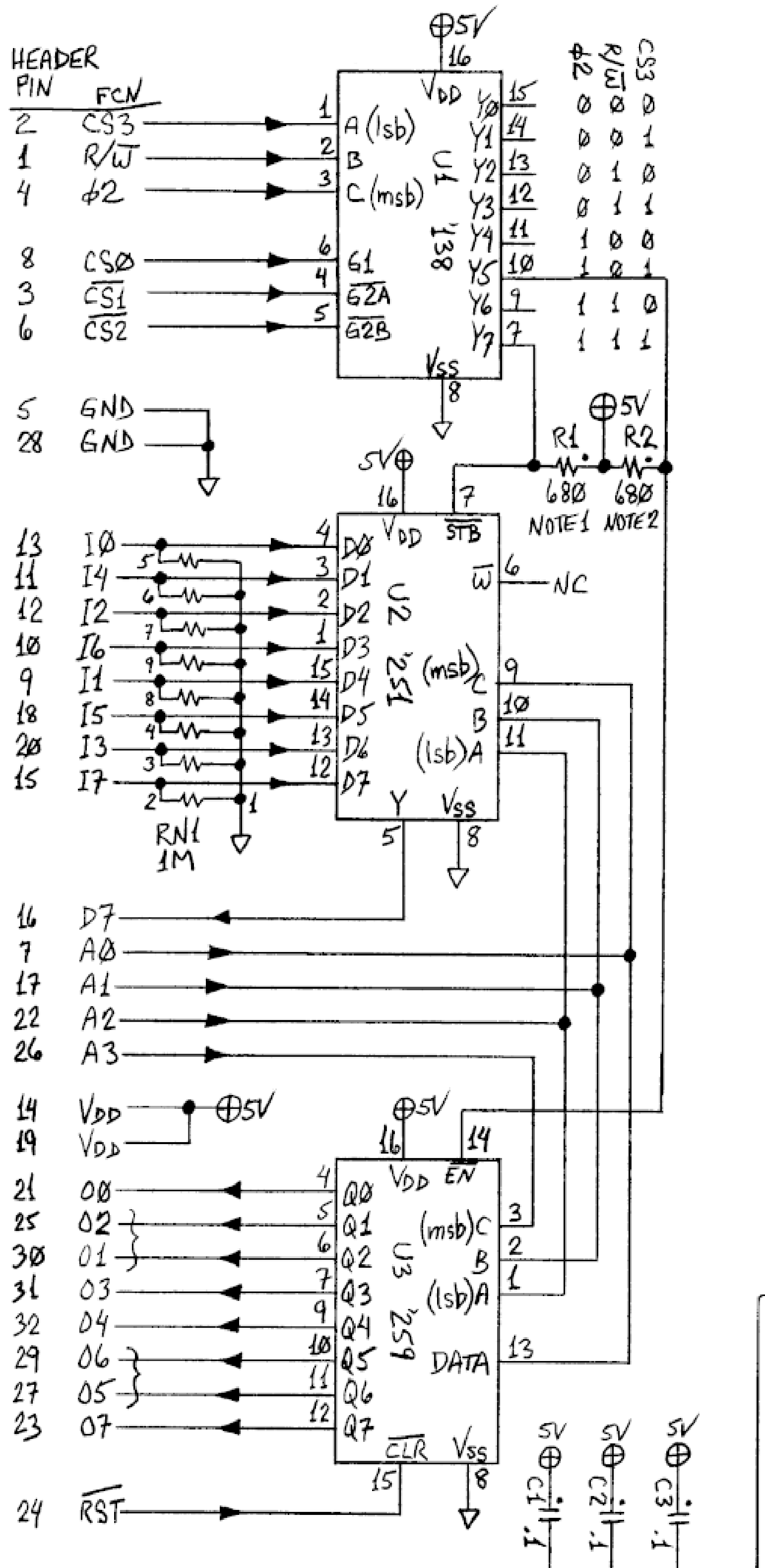
For reading from the 74xx251, the address selects which bit (out of 8) to put on D7 which the 6502's `BIT` instruction transfers to the N flag, or D6 which `BIT` transfers to the V flag, depending on your connections. The data line is high-impedance when the device is not being read.

Neither reading nor writing needs or affects the accumulator or X or Y. Reading will only affect the status register. Writing won't affect any register. Single-bit I/O becomes more efficient than it is on a VIA.

65c02 assembly-language example source code follows, on page 4.

```
BitIOaddr:  EQU  $6100          ; (Replace address as necessary.)
Bit0outLo:  EQU  BitIOaddr+0    ; Remember the low addr bit tells
Bit0outHi:  EQU  BitIOaddr+1    ; whether to make the output bit low
Bit1outLo:  EQU  BitIOaddr+2    ; (0) or high (1).  The bit number
Bit1outHi:  EQU  BitIOaddr+3    ; is in addr bits 1, 2, and 3.
Bit2outLo:  EQU  BitIOaddr+4
Bit2outHi:  EQU  BitIOaddr+5
 <etc.>
Bit7outLo:  EQU  BitIOaddr+$0E
Bit7outHi:  EQU  BitIOaddr+$0F


Bit0in:     EQU  BitIOaddr+0    ; These share addresses with the
Bit1in:     EQU  BitIOaddr+1    ; outputs above.  What makes the
Bit2in:     EQU  BitIOaddr+2    ; difference is the R/W line, low
 <etc.>                         ; for output and high for input.
Bit7in:     EQU  BitIOaddr+7


; You can of course give them additional names suitable for the
; application, such as:

SPIclkLo:   EQU  Bit0outLo      ; STA addr to put SPI clock line low.
SPIclkHi:   EQU  Bit0outHi      ; STA addr to put SPI clock line high.
MOSI_Lo:    EQU  Bit1outLo      ; Same with Master-Out Slave-In line.
MOSI_Hi:    EQU  Bit1outHi
MISO:       EQU  Bit1in         ; Addr for testing Master-In Slave-Out line.
SEL_ADC:    EQU  Bit2outLo      ; Addr to select analog-to-digital converter.
DESEL_ADC:  EQU  Bit2outHi      ; (The ADC is fast, but not quite fast enough to
                                ; put on the µP bus, so it goes on a VIA port.)


; To make an output bit low, write to it with the addr lsb=0:

     STA  SEL_ADC       ; STX, STY, or STZ works just as well.
                        ; The register contents are irrelevant.


; For an example to contrast the efficiency, the SPI bit-bang code at
; http://wilsonminesco.com/6502primer/SPI.ASM uses LDA #2, TSB VIA3PB
; for MOSI_UP, taking 8 cycles and not preserving the accumulator, and
; taking a bit from a VIA port which you might have wanted to keep
; intact for something that needed all 8 bits together; whereas with
; the bit-I/O module, it's only STA MOSI_Hi, 4 cycles, and the other
; penalties are non-existent.


; To test an input, use the BIT instruction which, regardless of what
; was in any register, makes the N flag reflect what was in data bit 7,
; or the V flag what was in bit 6:

     BIT  MISO          ; Follow BIT with a
     BMI  <label>       ; conditional branch.

; Note that there's no ANDing or ORing necessary, no read-modify-write,
; and there's no dependence or effect on A, X, or Y.
```

Schematic:

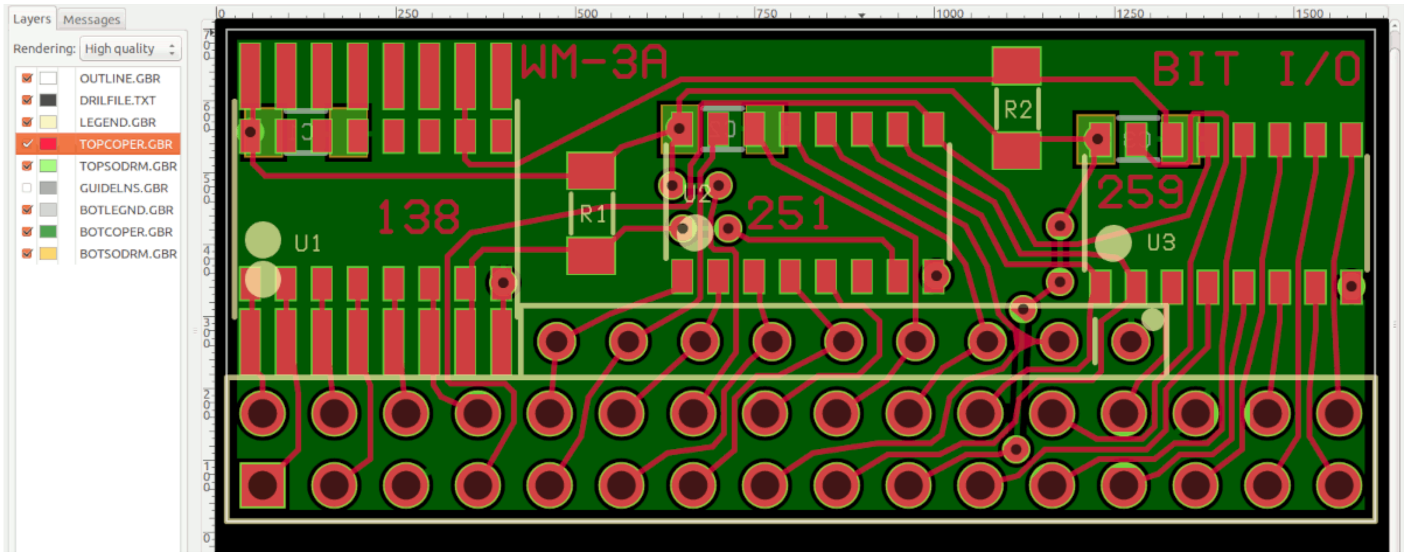


FUNCTION	PIN #	PIN HEADER		PIN #	FUNCTION
R/$\overline{W}$	1	▣	O	2	CS3
$\overline{CS1}$	3	O	O	4	φ2
GND	5	O	O	6	$\overline{CS2}$
AØ	7	O	O	8	CSØ
I1	9	O	O	1Ø	I6
I4	11	O	O	12	I2
IØ	13	O	O	14	VDD
I7	15	O	O	16	D7
A1	17	O	O	18	I5
VDD	19	O	O	2Ø	I3
OØ	21	O	O	22	A2
O7	23	O	O	24	$\overline{RST}$
O2	25	O	O	26	A3
O5	27	O	O	28	GND
O6	29	O	O	3Ø	O1
O3	31	O	O	32	O4

WM-3A BIT I/O MODULE
5/11/22 REVISION

NOTES:
1. INSTALL R1 IF U2 DOES NOT HAVE TTL INPUT LEVELS.
2. " R2 " U3 " " " " " "

Board layout:



ATTENTION
STATIC-SENSITIVE DEVICES
HANDLE ONLY AT
STATIC-SAFE WORK STATIONS

As with all CMOS parts, the ICs on the bit-I/O module are static-sensitive. Please observe ESD handling precautions. For more information, see: http://ics.nxp.com/packaging/handbook/pdf/pkgchapter3.pdf or http://www.zarlink.com/zarlink/esd-appnote.pdf or do a web search for "electrostatic discharge" or "ESD handling precautions". You will get tons of results. Many of the handling measures may seem extreme and even cost-prohibitive for a private individual to do his own construction. It is possible to handle ESD-sensitive parts without going to extreme lengths *if you understand what does the damage and constantly keep it in mind* as you handle the parts. Dropping your guard can result in damage in an instant. For equipment, consider a grounded anti-static mat on the workbench to be a minimum, and keep skin (like a bare forearm) in contact with the mat at all times while handling static-sensitive parts. For minimal cost, an anti-static wrist strap (connected to something grounded of course) goes a long way, removing the requirement to always be in contact with the mat to discharge static. The mat and the strap should be connected to something grounded.

Disclaimer: Garth Wilson and Wilson Mines Co. will not assume any liability arising out of the application or use or misuse of any product or circuit described herein.

Wilson Mines Co., Whittier, CA
WilsonMinesCo.com wilsonmines@dslextreme.com

Rev 0, Aug 16, 2022