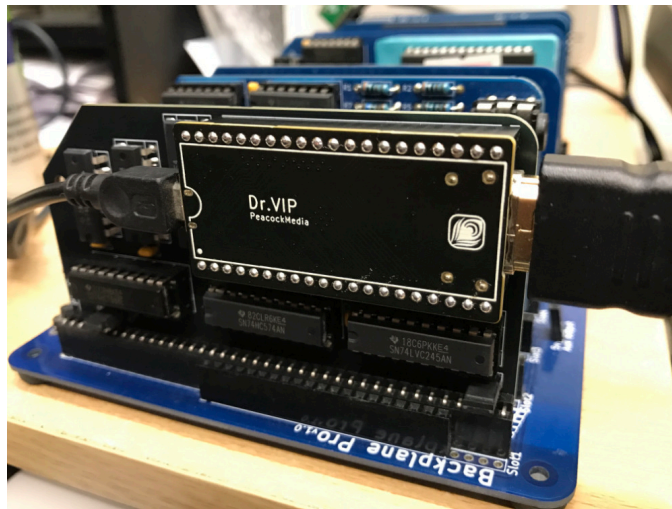


TMSEMU3 - TMS9918A emulator video card for RC2014



The TMS9918A is a very capable video chip and I feel that graphics are an essential addition to an RC2014.

This module emulates the TMS9918A chip and adds HDMI-compatible (DVI) out, optional USB keyboard input and the ability to display the computer's serial output.

As from TMSEMU3 (marked r5.2 on the back of the board) it has an 80-column colour terminal display, with the terminal and TMS emulations being switchable.

CP/M RC2014 Pro and Zed Pro

I recommend MSX ports (\$98/\$99) but the Tatung Einstein ports (EIN) 08/09 should work fine. The examples that I have built should detect the card at either position.

Note that all of the examples I've built leave out the detection of port \$Bx (Colecovision) as this conflicts with the ram/rom paging in the RC2014 CPM setup. (Note that J B Langston's code as currently published unfortunately probes this port first.) Using Colecovision ports should be fine with ROMWBW machines.

Note that on a real ColecoVision, the entire A0-BF range is assigned to the TMS9918A, but this module uses BE and BF. I believe this is fine because (from J B Langston's documentation) "all known games only use ports BE and BF".

32k classic RC2014

Jumper to Tatung Einstein ports 08/09 (EIN). If you're using the ACIA 68B50 serial module then the MSX ports \$98/\$99 will conflict.

Seeing the serial output of the computer

The **VDU jumper** connects or disconnects the bus TX line. With this in place, the module will display the computer's serial output from startup* and thus allow you to use your computer without using another terminal.

***Note** that the TMSEMU may take a little longer to initialise than the computer, it isn't quick enough to capture the SCM welcome line generated on classic and CP/M RC2014s. When you type, you should see your typing and the computer's output as expected.

By default, from startup, the module will be in 80-column terminal mode, which will respond to the ANSI codes for colour, cursor positioning etc. If anything doesn't work as expected, please tell me.

It is possible to switch to TMS mode at any time, or lock TMS mode from startup, or switch to terminal mode at any time or lock terminal mode from startup.

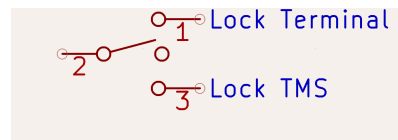
Switching mode

By default, if the module detects a activity on the TMS ports, it will switch to TMS mode. The examples written for RC2014+TMS generally perform a port detection before anything else, so the auto-switching will happen and the TMS emulator should then respond as expected.

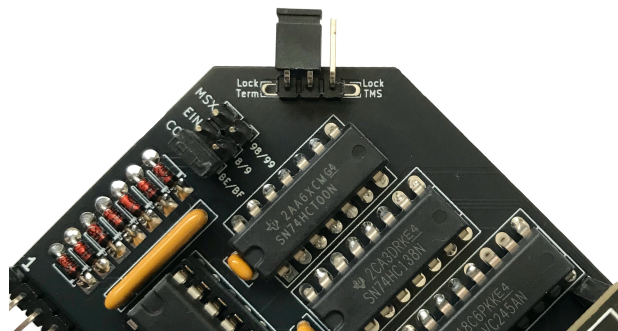
If software sends important data to the TMS ports while the module is still in terminal emulation mode, some of this data may be lost and the software won't work as expected. (The MSX8 system is a good example). In these cases, it'll be necessary to trigger a switch to the TMS mode before starting the application.

You could do this with a small CPM program which simply reads one of the TMS ports. You'll be able to type its name to trigger the change (eg A:TMS if you've named it TMS.COM and put it in drive A:)

Alternatively the module has pads for a physical switch or jumper, which could be a 3-position switch for "lock terminal", "lock TMS" or "auto". The switch / jumper needs to short the centre pin (ground) to one or other of the pads, or neither of them (centre position in the diagram below) for auto mode.



This footprint (by the chamfer) is **deliberately left unpopulated** when I supply the module. You may want to solder in header pins (supplied) for a jumper, or solder a 3-way slide or toggle switch, or solder wires which go to a switch elsewhere. (The elongated pads allow for a switch with three pins that are at 0.1"pitch or more.)



When in TMS text mode and when the VDU jumper is in place, you should still see terminal output. This uses the TMS text mode which is 40-column and has just a background/foreground colour combination. The terminal functionality when in TMS mode is limited. It has some wrapping and scrolling and a few other functions, but not the full implementation of the escape codes (cursor positioning, reverse video, colour etc.) To achieve this, the TMS emulator's VRAM is pre-populated with a memory map and 6-bit font.

If the program you run uses a different screen mode, you may not be able to see terminal output after breaking the program. Terminal output will only be displayed when the TMS emulator is in text mode. It's your program's responsibility to switch the TMS back to text mode before quitting. This is pretty unlikely unless you've added that yourself and so you may have to reset your computer after running a TMS program. Using your switch to "Lock terminal" and then "Lock TMS" may well work.

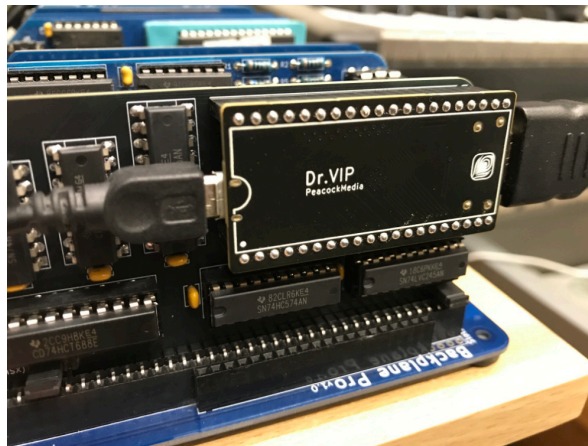
You can remove the VDU jumper to disconnect the module from the serial output.

Using a USB Keyboard

This is a new feature since TMSEMU2 (late August 2024). It is a hardware update, so there is no firmware update to give this feature on earlier TMSEMUs.

Plug your keyboard into the micro USB socket on the Dr.VIP board (opposite end from the HDMI socket). This is the reason that three chips are not socketed.

You will need to use an 'OTG' or on-the-go adaptor, which allows you to plug a USBA plug into the micro-USB socket.



The screen will turn black for a second or two. This is expected when the keyboard mounts. Then the screen should return to its previous state.

The module's connection to the serial line (RX on the bus) is managed by the module. Until a USB keyboard is mounted, it'll be disconnected and your usual serial input (ie from a terminal connected to the serial module) should work as normal. With the keyboard mounted, TMSEMU is connected to the computer's RX line and it should send characters from your keyboard. You may find that data sent via any other terminal may not work while the USB keyboard is mounted. This is expected. You can use one or the other.

The keyboard mapping is built into the firmware. If you find that the mapping isn't correct for your country please contact me.

<add links here to firmware, with instructions for updating>

The Interrupt line

The real TMS chip can generate an interrupt signal which coincides with the vertical blank, ie 60 Hz. This is software controlled. It does nothing until enabled and when it asserts, then a status port read is necessary to reset it. See the TMS9918A datasheet for more information (link below **Going further**). It's connected to the bus INT line, but the TMSEMU keeps it disconnected (hi-z state) unless it is actually asserting.

The Status register

The highest bit of the status register is the 'frame' bit which is reset to 0 when the status register is read, and is set to 1 at the vertical blank point. This allows you to detect the vertical blank for timing games and drawing to the screen without tearing.

Another bit in that register is the sprite collision bit, used by some games. TMSEMU3 correctly provides this flag.

Other bits in the status register are concerned with the 'fifth sprite'. A limitation of the real TMS chip is that it can display a max of 4 sprites on one scanline. TMSEMU3 doesn't have this limitation, and so it currently doesn't supply the 'fifth sprite' information in the status register.

Configuration

Since firmware version 5.2.3 it is possible to access on-screen configuration which allows you to choose:

- Keymap (UK / international)
- 24/48 rows
- Default text colour (white / amber / lt. green / dk. green)

To access the configuration screen, plug in a USB keyboard as described earlier. If possible, use the 'lock Term' jumper or switch setting. In terminal mode, use the keystroke: right alt+c (it has to be the right-hand alt key). For recent revisions of the board, a physical button triggers the configuration screen.

The configuration screen shows the keystrokes for changing the settings. The [s] key will save the settings. If you have 'lock Term' jumpered, then the module should reset itself and load the new settings.

48 rows is useful for some games or seeing more information on the screen at once. You may find 24 easier on the eye.

Going further

If you want to program the TMS chip, the documentation is here:

<https://peacockmedia.software/tms/TMS9918datasheet.pdf>

Troubleshooting

If you experience issues, please feel free to contact me at shiela@peacockmedia.co.uk

Software

My own collection of J B Langston's examples and my own game ports and other software is here: <https://github.com/shieladixon/RC2014-TMS9918A>