USB Serial Teletype Loop Adapter
mfg. by Eric Volpe for Greenkeys subscribers, Apr. 2016. (contact epvgk@limpoc.com for info)

This is a USB to teletype current loop interface. It connects to a computer running Windows, Mac OSX, Linux, Android, or other operating systems via a USB Mini-B connector, and presents itself as a USB serial port, while the other end connects directly in series with a teletype loop, and relays data bidirectionally between the two. It does not provide loop current itself, so a loop power supply with fuse is needed if not already present. For more info, http://heepy.net/index.php/USB-teletype

Installation Instructions:

Linux / BSD:
No driver installation needed. When connected, the device will appear as "*/dev/ttyACM0*" or similar and can be opened with any terminal program or anything that works with a serial or USB terminal.

Apple Mac OSX:
No driver installation needed. The device will appear as "*/dev/cu.usbmodem1411*" or similar.

Windows:
Windows may require an information (INF) file to use the device. With the device disconnected, download this file from http://teletweety.com/usbtty.inf, save it to a file, and install it by right clicking the icon and selecting "Install." Confirm any dialog asking if you're sure. Then connect the board. It will appear as a "*COM:"* device; you can look in the Device Manager for "Teletype Adapter" under "Ports (COM & LPT)" to see which COM number it was assigned.

Use:
When powered up and not connected to a current loop, both the green and orange LEDs should be lit. The **orange** LED displays the output (toward the loop) – illuminated means it is in the marking state, and off means it has opened the circuit. The **green** LED shows the input – it is lit when the loop is open, and off when the loop has current. This way it's more fun to watch when you're typing.

There are two blue terminal blocks, marked "RX_LOOP" and "TX_LOOP" - as shipped, the device is set up for a single loop, and there is a jumper installed between the two. Connect the board in series with your existing loop using the two open screw terminals. Polarity does not matter. With current flowing in the loop, the green LED should go out.

To verify operation, open a serial terminal program on your computer, and open the serial device name associated with the teletype adapter. On windows this will be "*COM#:*" where # is the number determined from the Device Manager above. On OSX, Linux, or BSD, it will be the "*/dev/XXX"* file that appears when the device is connected, which can also be found by looking at the output of the "dmesg" command. The other settings configured in the terminal program will be ignored, so they can be left at the defaults.

Typing characters on the terminal program should result in flickering of the orange LED, and if it is connected to the loop, it will also print the text and flicker the green LED as the characters are simultaneously received. If you have a machine with a keyboard, text typed on it will also print to the terminal window. From the point of view of the teletype loop, your desktop computer is now just another Teletype machine on the loop. If you send a break with the terminal program, the board will break the loop current for 0.5 seconds; if you hit the BREAK key on a machine or otherwise open the loop, the board will print "[BREAK]" to the terminal window when the loop current returns. (This behavior is configurable, see **Configuration** section.)

Configuration:
All setup and configuration is done via the terminal window from your computer; it ignores attempts by the host operating system to configure speed, etc. To enter configuration mode, press the button on the board next to the USB port, marked "**SW1**". At this point the device will stop listening to the teletype loop and will present a list of available commands and a "*cmd*> " prompt to the terminal window. At this point, type "*show*" to see the complete set of both active and saved settings. To enable a yes/no option, just type the name of the option, such as "*showbreak*" or "*autocr*". To disable it, prefix it with "*no*", for example "*noshowbreak"* or "*noautocr*". To set an option that takes a value, such as baud rate, type the configuration option name followed by the desired value.

Note: all settings take effect immediately, but are reset on power-up unless saved to permanent memory. To save the settings, type "*save*" and they will become the new defaults loaded on power-up. If you want to reset to "factory" defaults,

use the command "*eewipe*" and then disconnect and reconnect the board. The factory settings are: 45 baud, 5-level code, ASCII translation active, always send CR/LF, no unshift on space, display BREAK, and no auto-CR at end of line.

Configuration Settings:

translate: (default: enabled)
When enabled, characters send from the host computer are translated from ASCII to ITA-2 5-level code before being sent to the loop; likewise, ITA-2 5-level characters received from the loop are translated into ASCII before being sent to the host computer. ASCII characters for which there is no ITA-2 equivalent are silently ignored. When disabled, no translation is done; the low 5 bits of each character from the host are sent intact as a 5 bit characters to the loop, and characters received from the loop are passed intact to the host as 8 bit characters with the top three bits set to 0. This mode is meant for use with software that prefers to do the ASCII/ITA-2 translation itself, such as HeavyMetal or BaudotRSS. (Note, however, that this firmware also allows multiple and changeable translation tables – see "**Extras**" section.)

crlf: (default: enabled)
When enabled, either CR or LF from the host will be sent to the loop as a CR and then an LF. Different operating systems and terminal programs handle CR and LF differently, and this makes sure the loop gets both. When disabled, CR and LF are handled independently. You probably only need to disable this option if you want to print overstrike pictures.

autocr: (default: disabled)
When enabled, if a line of text from the host exceeds 70 characters, a CR and LF will be sent to the loop before any more characters from the host are printed. This is intended as a failsafe to prevent overstrike at the end of lines. When disabled, no intervention happens to prevent overstrike. Note that this option counts printable characters starting from the most recent CR, so it has to see at least one CR before it begins to function.

usos: (default: disabled)
When enabled, a space character received from the loop will shift the receiver into LTRS mode, the same as if LTRS were received. When disabled, receiving a space does not cause any shift.

showbreak: (default: enabled)
When enabled, a break received from the loop will send the text "[BREAK]" to the host computer. When disabled, no special handling of breaks occurs. Note that disconnecting and reconnecting the loop is indistinguishable from a "break" and will also display "[BREAK]".

baud: (default: 45.45)
This sets the data rate for both sending and receiving to the specified number of bits per second. The value given must be an integer, but if it is one of the commonly used values, it will be adjusted to the common exact speed, for instance "*baud 45*" will actually set the speed to 45.45 baud. Values of 45, 50, 56, 75, and 110 will be set using timing I measured using a scope; others will be calculated on the fly but should be good enough. The minimum speed is "1" and will test your patience; the maximum is governed by a number of factors such as the inductance present on your loop, the loop current, and the limits of the optical isolators on the board; I've had it working up to about 1000 baud, but any speed reasonable for a mechanical TTY will be fine.

exit:
This leaves configuration mode and returns to relaying data using the current settings. If you want the currently active settings to be used next time the device is powered up, make sure to use the "*save*" command first to save them to non-volatile memory.

Terminal programs:
Windows: a nice writeup is at http://learn.sparkfun.com/tutorials/terminal-basics/hyperterminal-windows
Mac OSX: "screen" also works on OSX, and the Brew and Fink package systems also supply "minicom" and "c-kermit"
Linux: I like to use "screen /dev/ttyACM0 300", but there is also "cu", "minicom", "ckermit", or "socat".
For more elaborate functionality, ***HeavyMetal*** (http://www.albinarrate.com/heavymetal.html) or ***BaudotRSS*** (https://github.com/John-Nagle/baudotrss)

Extras: These are features which are either experimental or less likely to be of interest to the average user.

8-bit mode: (experimental)

To try to use the board to talk to an ASCII machine like a Model 33, enter config mode and set "*8bit*" and "*baud 110*" - this will also change various other options for you in order to hopefully make it compatible with 110 baud ASCII machines and old computers. Since as far as I know most such application are full-duplex, you may need to remove the jumper between the TX and RX terminal blocks and connect each block to its own loop. In this mode there is no character translation, and the transmit and receive systems are switched to 8 bit instead of 5 bit. Also, since again as far as I know most ASCII machines run on a 20mA rather than 60mA loop, you will be operating near the lower end of the range that the receiving circuit can sense reliably, though it did work OK for me in testing. If you have problems receiving, slightly increasing the loop current may help. If it falls below 15 mA it will almost certainly not work.

Alternate translation tables:

The default translation between ASCII and ITA-2 / Baudot is probably sufficient for most uses. The tables are read from nonvolatile storage in real time as characters are sent and received. It is possible to modify the provided tables or to add up to 5 additional tables, which can be selected using the "***table***" command. The default table is 0; additional user-added tables start at 1. To add a new table, you will need to use the "***eewrite***" command to update values stored in nonvolatile memory at the correct addresses.

Each table consists of 64 bytes; 32 bytes for LTRS followed by 32 bytes for FIGS. The same tables are used for both "sending and receiving. When a character is received from the loop, that character's offset is looked up in either the LTRS or FIGS table, and the corresponding value found at that address is sent to the host as ASCII. So for instance if the shift state is LTRS and the device receives 01010 (hex 0A, decimal 10), it will look at offset +10 from the beginning of the current LTRS table, find the value 0x52, and send it to the host, which will see it as ASCII 'R'. If the host sends the ASCII letter 'Y', the current LTRS table will be sequentially searched for a location whose value is 0x59 ('Y') and that location's offset (21) will be sent to the loop as 10101, where it will print as 'Y'. The default table (0) begins in eeprom at offset **0x0080** for LTRS and **0x00A0** for FIGS; table 1, if present, should start at **0x00C0** for LTRS and **0x00E0** for FIGS. When the "***table***" configuration setting is set to something other than 0, the value is multipled by 64 and used as an offset from the beginning of the first table. There's no bounds checking so setting nonsense values is likely to produce nonsense data.

To see the current contents of nonvolatile memory, use the "***eedump***" command.

To create or modify a table, use "***eewrite*** XXXX AA BB CC DD EE …." XXXX is the start address, and must be exactly four hex digits, padded with leading zeros. Subsequent arguments must be exactly two hex digits each, separated by exactly one space, and will be written to eeprom locations starting at XXXX and increasing one at a time. Since the commandline input buffer is small, you can only write 16 values at a time. The format is exactly the same as the output of "***eedump***". Note also that the "***eewrite***" command has no bounds or format checking and will allow you to corrupt all parts of nonvolatile memory, including the baud rate and option settings starting at location 0. It can not, however, modify program memory, so there is no risk of irrevocably damaging the device (see Factory Reset below.)

If you don't like messing with hex, send me an email indicating what you want to change in the translation tables, and I'll send you lines of text you can cut and paste to the board in config mode.

Autoprint: (experimental)

Run the "***automsg***" command to store a text message up to 512 bytes in eeprom. Then enable both "**showbreak**" and "**autoprint**" modes. The text can be printed by pushing a break key or otherwise briefly interrupting loop current. This will not currently work in 8bit or notranslate modes. It may be finicky. It will work even with a dumb power supply rather than a computer powering the adapter. Leave it turned off unless you really want this.

Factory Reset:
Lastly, if you find you've messed up the configuration in some way and want to start from scratch with the configuration exactly as it was when shippped, run the command "***eewipe***" and then disconnect and reconnect the board. Note there is no "are you sure?" confirmation.

Warnings and other information:

The board can handle loop current from 20mA to 100mA and voltages up to 200v. Inductive spikes from the loop are snubbed by a transient voltage suppressor to around 250v, which protects the output optoisolator from damage under reasonable conditions. The receive loop current is sampled by a 2 watt, 150 ohm resistor placed in series with the loop, which **will become warm** under normal use. Exceeding the recommnded loop current may exceed this resistor's wattage rating. If you need to accommodate unusual loop configurations, you can replace this resistor.

Every effort has been to design and manufacture the board so as to isolate high voltage portions of the circuit from the computer interface, but this is a component for experimenters, not a consumer electronics product.

There is potentially hazardous voltage present on the loop interface side of the board when connected to a loop. In addition to the shock hazard, if using the device without an enclosure, never allow the board to contact a conductive surface or object as this could breach the optical isolation between the loop interface side and the computer interface side of the board, potentially causing severe damage to the host computer.

This item is provided in the hope that it will be useful but without any warranty of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to use of this item lies with the purchaser. In no respect shall the seller incur any liability for any damages, including, but not limited to, direct, indirect, special, or consequential damages arising out of, resulting from, or in any way connected to the use of the item.