Serial debug with Tera Term

What? Why?

Embedded devices rarely come with a screen and even if they do, it's not to provide the developer with feedback.

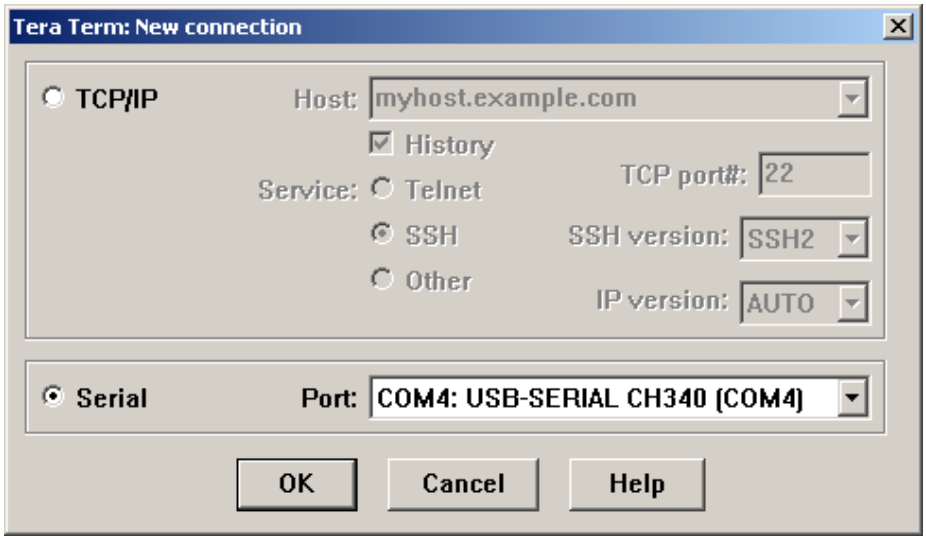
So we most commonly put statements in to our code to send text over a serial port which we can view on our development workstation.

Typically we use printf to send text with the value of variables. How you setup the serial port varies between device manufacturers but normally we only need to set the speed or baud rate. Many systems default to 9600 which by todays standards is quite slow, 115200 has a good turn of speed that won't hold up the running code without taxing the MCU to keep that rate going.

Tera Term

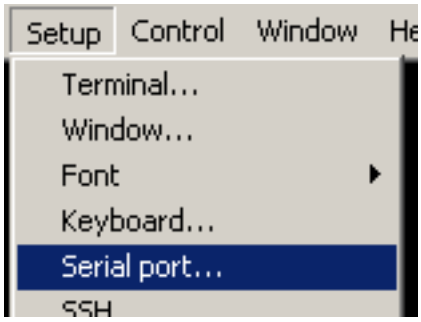
Tera Term has provided a serial terminal since 1996, so it has many features but as an open source project isn't exactly a work of art.

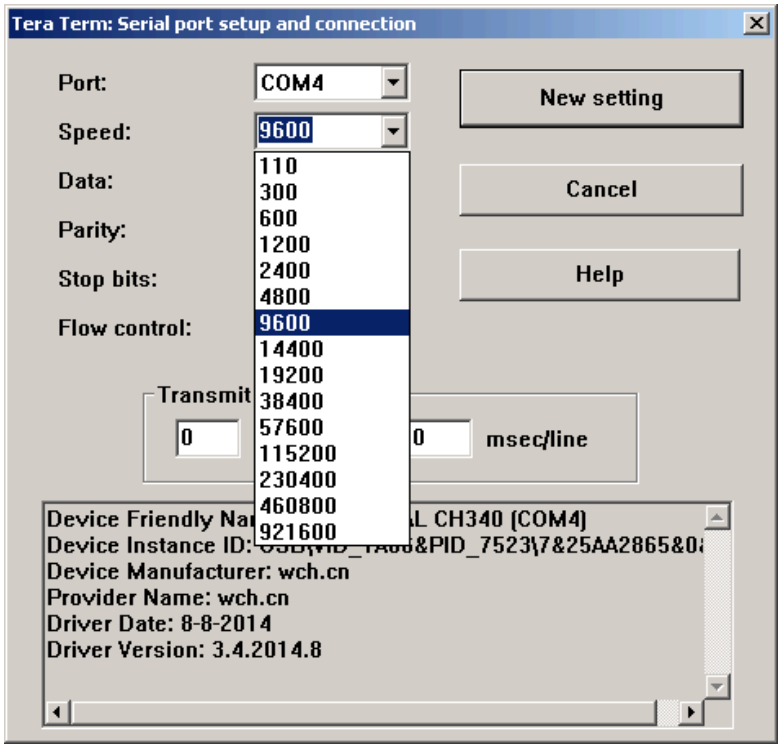
When you start it up, it will ask you to select between a network or a serial connection:



You choose the serial port - quite often the last one on the list.

To set the baud rate, from the `Setup` menu choose `Serial port …`





You can change the size of the typeface from the `Font` sub-menu on the `Font` menu item on the `Setup` menu

You can change the colours in the `Window` menu item.

Carriage Return and/or Line Feed?

Or is it New Line?

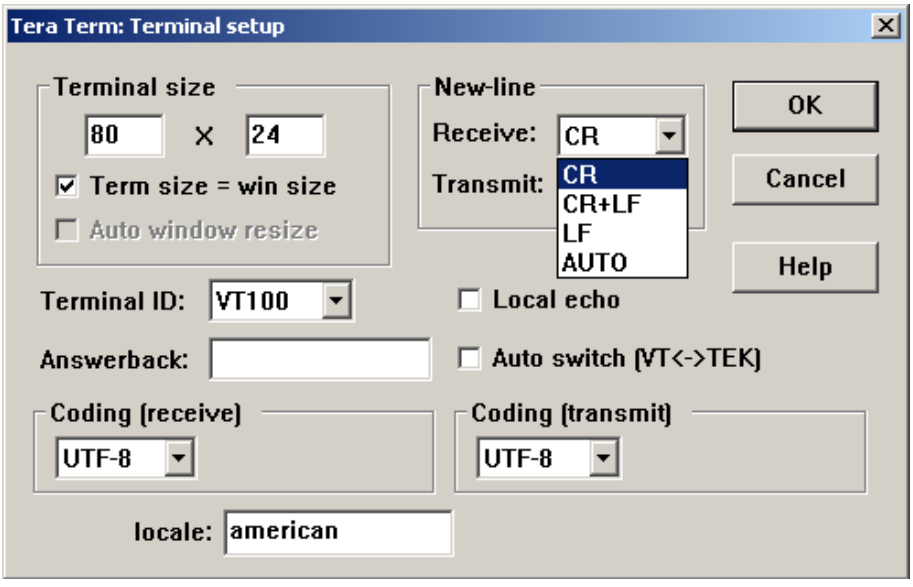
In the beginning computers used a teleprinter for output - an automated typewriter - that had to be told to return the type head to the beginning of the print carriage and then feed the paper up one line.

When computer screens were introduced the commands were reduced to just one, usually “New Line” aka LF.

Some operating systems used “Return” aka CR, principally Macintosh System pre OS-X.

Windows settled on Return / New Line aka Carriage Return - Line Feed aka CR LF.

Tera Term can cope with any combination of the above by setting the New Line option in the `Terminal` menu option to Auto.



Printf and its options

The printf function takes a string with markers and then as many arguments / variables as there are markers. The markers all start % with a wide range of options. If you need to use %, you use two - %%.

You also specify the line ending you want to use. Almost all embedded C development uses New Line. Some code bases use CRLF to accommodate Windows terminal programs, but as seen above, this is not necessary. Most other platforms just ignore the CR.

Basic message:

```
printf(“This is a sample message with a new line at the end of it\n”);
```

The New Line is the \n. The \ indicates that the next character is a command code. If you need to use a backslash, you use two - \\

If you need to use CRLF, it would be \r\n.

Message with a variable:

```
printf(“Important variable = %u\n”, theVariable);
```

The u format is for an unsigned integer.

Message with two variables:

```
printf(“Variable1 = %d, Variable2 = %x\n”, variable1, variable2);
```

The d is for a signed integer and the x is for an unsigned integer shown in hexadecimal.

Format codes:

d or i	Signed decimal integer (16 bits)
u	Unsigned decimal integer
x	Unsigned hexadecimal integer
f	Decimal floating point
e	Scientific notation (mantissa/exponent)
c	Character
s	String of characters

Upper case X and E will use uppercase letters.

There are prefixes for bytes which is hh and l for 32bit values and ll for 64bit.

So a %hhu would be an unsigned byte and %ld would be a signed 32bit value.

You can set the width with a number in front of the the format codes. So %4d would right align the value 4 characters wide. Spaces are used to pad the left hand side as required.

You can set the decimal places for a float with a decimal point. So %10.4f would be a floating point number 10 characters wide with 4 decimal places.

You can pad with 0’s instead of spaces with a 0 in front of the width. So %04X would format an integer with leading 0s - so 3715 (decimal) would appear 0E83. If you put 0x before the %, so you have 0x%04X, then it will appear 0x0E83. There is the # prefix that appears to do this, but it doesn’t display for zero values so isn’t that useful.