**Tweetwronger**

The original idea was to connect a teletype machine to twitter for the Typewronger bookshop and typewriter emporium in Edinburgh, Scotland. The main problem was obtaining a teletype machine.

A number of daisy wheel electronic typewriters produced in the 1980s had computer connection options as the only general printers available were dot matrix and for professional use a “letter quality” printer was prefered which tended to be a daisy wheel printer. Since the hardware for a daisy wheel typewriter incorporates a daisy wheel printer the option to use one machine to do two tasks was a great saving to customers. One of these was an obvious substitute for a teletype machine.

Some research online suggests that during this period Nakajima of Japan made electronic typewriters that were branded by many different companies. The particular unit used for this project claims to be made in Western Germany but possibly only had final assembly performed there or was based on a japanese design. Therefore further research should be carried out but given the cross pollination at this time the details here may be relevant to other typewriters branded by other manufacturers.

A Triumph Adler SE325 came up on eBay which has an 8 pin DIN connector on the back that was originally used to connect to the TA IFD1 interface box that would allow it to connect to a computer over serial or parallel. The IFD1 talked to the typewriter over a serial connection with 5V levels but the protocol for this communication was a proprietary one over which very low level commands were sent. Fortunately an article from “ST Computer” magazine in Germany from July 1988 explains much of this protocol and can be found online at :

<http://www.stcarchiv.de/stc1988/07/gabriele-9009-1>

and:

<http://www.stcarchiv.de/stc1988/08/gabriele-9009-2>

It is possible that other commands exist other than those in the article but either finding a service manual or a decompilation of the ROM is likely the the only way these will be found.

The article provides the pinout of the 8 pin DIN connector, the settings for the serial communication and the codes needed to make the typewriter work

Pinout:

1: 35V

2: DTR

3: DSR

4: 5V

5: RX

6: 10V

7: TX

8: 0V

(Voltages are supplies an can be ignored)

Serial port configuration:

4800baud

1 start bit

8 data bits

1 stop bit

Flow control uses the DSR and DTR lines. The specifics of how they are used for flow control are described below:

In general operation the typewriter appears to have a buffer of probably 32 commands (each 2 bytes). When a byte is sent there is roughly 350 microsecond delay after which the DTR pin goes high for approximately 1 millisecond to indicate that the typewriter has received it and is ready for the next byte which must start after this pulse. If the buffer is full this signal is delayed until space becomes available. For this reason the computer must monitor the DTR line and moderate its transmission appropriately. Very little data is transmitted from the typewriter with any of the known commands but when it is the DSR line must be high to allow the typewriter to transmit the data.

For this implementation a USB to 5v serial cable based on FTDI FT232RL was used and the #CTS line connected to DTR and #RTS connected to DSR. One problem with this configuration is that the computer must catch the 1ms acknowledge pulses which naturally cannot be guaranteed with a programme running on a time sliced operating system. The solution was to put flip-flop in between the #CTS and DTR lines with reset connected to the #RTS/DSR line. For simplicity and based on availability, this was in the form of a Digispark (Attiny85 based board that can be programmed with the Arduino development environment) running the following code in C++:

//digispark code to act as a latch for the TW ready line so that the host //computer never misses it.

const int input = 0;

const int output = 1;

const int rset = 2;

void setup() {

pinMode(input, INPUT);

pinMode(output, OUTPUT);

pinMode(rset, INPUT);}

void loop() {

if (digitalRead(input) == HIGH)

{

digitalWrite(output, HIGH);

}

if (digitalRead(rset) == HIGH and digitalRead(input) == LOW)

{

digitalWrite(output, LOW);

}

}

This latches the output to 1 when the input is 1 and only resets to 0 if the input is 0 and the the rest pin is 1

Once a byte is sent the the computer waits for the #CTS pin to go high, then after 5ms sets #RTS high then low and checks again and repeats until the pin goes low. Python code for data send subroutine:

def TWsend(data):

for byte in data:

n = 200

while not port.cts :

time.sleep(0.005)

n -= 1

if n < 0:

raise NameError('Port not responding')

port.write(byte)

n=200

while port.cts :

time.sleep(0.005)

n -= 1

if n < 0:

raise NameError('Port not responding')

n = 200

while not port.cts:

time.sleep(0.005)

n -= 1

if n < 0:

raise NameError('Port not responding')

port.rts = False

port.rts = True

(note: the pins on the cable are logically inverted)

Each command for the typewriter is made up of 2 bytes. The communication commands for the typewriter given in the ST magazine article are as follows:

0xA0, 0x00 : CLEAR -> the typewriter goes into OFFLINE state and becomes "typewriter" again.

0xA1, 0x00 : START -> transition to ONLINE is being prepared

0xA2, 0x00 : STX -> typewriter becomes "printer", state "ONLINE", transmission of print commands can begin

0xA3, 0x00 : ETX -> Data transfer is interrupted and waiting for continuation

0xA4, 0x00 : ENQ -> typewriter should report status

On this model of typewriter the TW/M key must be pressed and then option 7 selected to go into online mode. Once this is done the typewriter waits for the DSR then sends 0x01. After this the following commands must be sent to bring it online:

0xA0, 0x00

0xA1, 0x00

0xA4, 0x00

0xA2, 0x00

To return it to offline:

0xA3, 0x00

0xA0, 0x00

After the typewriter is connected and online then the following commands can be issued to control the typewriter

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| write character | | | | | | | | | | | | | | | |
| Byte 1 | | | | | | | | Byte 2 | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 0 | typewheel position 0 to 100 | | | | | | | Move right/left  After strike | suppress move |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| set width | | | | | | | | | | | | | | | |
| Byte 1 | | | | | | | | Byte 2 | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | width in 1/120” steps | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Reset | | | | | | | | | | | | | | | |
| Byte 1 | | | | | | | | Byte 2 | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ribbon motor | type motor | carriage motor | 1 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| space | | | | | | | | | | | | | | | |
| Byte 1 | | | | | | | | Byte 2 | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | number of 1/120” steps. If zero uses current width | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| backspace | | | | | | | | | | | | | | | |
| Byte 1 | | | | | | | | Byte 2 | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | number of 1/120” steps. If zero uses current width | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Move | | | | | | | | | | | | | | | |
| Byte 1 | | | | | | | | Byte 2 | | | | | | | |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| 1 | 1 | (right/left)  (up/down) | horizontal/  Vertical | high nibble of movement  in 1/120” steps for horizontal and 1/96” for vertical | | | | low byte of movement in  1/120” steps for horizontal and 1/96” for vertical | | | | | | | |

Potentially there are other commands that may exist ( 0b10000010 and 0b10000101 to 0b10111111 less 0b10100011) that we do not have documentation for.

To control the typewriter a set of python functions was written that wraps up each of these functions and provides limited input sanitation, these are found in the typecontrol.py file.

The typewriter does not accept ASCII characters for printing but rather uses the position on the typewheel to specify the character to type. The wheel has 100 positions and this scheme means that the control system is alphabet independant. Currently a map of the installed wheel is hard coded into the python wrapper but really should be changeable.

To expose these to other processes, a simple web app was built with flask. This is sub optimal as it uses the flask server that should not be used for production due to scaling issues but as this system will be running only on a private wifi network with at most a couple of connections to it at once this should not pose any issues.

The flask web app exposes a simple API which can fully be explored by connecting to the IP address or hostname of the machine on which it is running on port 80.

The API uses HTTP GET requests to trigger the wrapped functions.

|  |  |  |  |
| --- | --- | --- | --- |
| call | description | variables  {} = optional | example |
| /connect | Connect to the typewriter and put it into online mode | port | 127.0.0.1/connect?port=’ttyUSB0 |
| /disconnect | Disconnect and return typewriter to typewriter mode | - | 127.0.0.1/disconnect |
| /move | Move the carriage and platten | {x}  {y} | 127.0.0.1/move?x=120&y=96 |
| /setwidth | Set the distance the carriage moves after typing a character.  If no width variable specified, reverts to 1/10” | {width} | 127.0.0.1/setwidth?width=10 |
| /type | Type text with bold and underline as specified. If wrap=true then text is wrapped to ]width if specified or 70 columns | {text}  {bold}  {underline}  {wrap  width} | 127.0.0.1/type?text=test&underline=true&bold=true&wrap=true&width=10 |

Other functions may be useful and could easily be implemented.