It Started So Well ...

Chris Hall

I took on the task of editing a series of books for the Signalling Record Society, the major part of each volume was a list of the signal boxes on each stretch of railway line. All I had to do was to present and index the data so that any entry could readily be found by the reader.

I wanted to make the information available on line, so I based my approach on creating an HTML master for both printing and viewing. The authors of the first volume had laid out the data (what type of signal box, opening and closing date, type of lever frame, etc.) in the form of a spreadsheet.

It therefore seemed obvious that I would need to read in each row, line by line, and hold the data in arrays. Within BASIC I could then produce an index and listing as a (very) large HTML page. Each entry was enclosed by HTML table tags.

The BASIC programme to do this (read a spreadsheet and produce an HTML page from it) grew rapidly (5500 lines in 2007, 9300 in 2009 and 15800 lines in 2015) as I added more features.

[Note: description of this programme is outside the scope of this magazine as it was written in BBC BASIC for Windows – mainly to avoid the 256byte limit on string length]

Initially the page was simply viewed in a browser window, navigating by means of 'hot links' but by suppressing the links and adding style sheets could be turned into a PDF using Acrobat Distiller. It was fairly easy to introduce photos to fill gaps at the bottom of pages and add page numbers — the result was acceptable.

I published the first volume in 2007: the book listed railway lines and signal boxes – their building design and the type of equipment they contained – and was rather dry: illustrations would certainly help. I was thus keen to illustrate the book with a coloured map for each section showing railway lines in the area covered, marking those listed in that section.

All I needed to do was to trace out each railway line in Britain, recording the grid reference and mile post mileage at each point. Then I would have the necessary data to produce a map.

I had, by now, found a web site (*npemap.org.uk*) that would display scanned map tiles from the 1940s (out of copyright) Ordnance Survey maps (showing old railway lines and lined up with the National Grid). I also had my mother's school atlas, showing the pre-Grouping railway companies (i.e. pre-1922) which would provide a suitable pastel-shaded backround for the maps.

At this stage I could thus see in a browser window the route taken by each railway line. I also knew the milepost mileage of each signal box and the milepost mileage at the extremity of each railway line.

Stages 1 and 2 would involve tracing each railway line and recording the result.

Back to the book

By painstaking manual methods, I had worked out the grid reference of each box in the first volume (published in 2007 with no maps) and the authors had provided the milepost mileage of each box (the distance along the line: all railway companies were required by law to fence their lines and to provide mileposts every quarter of a mile along every line).

However many boxes were listed with no grid reference quoted. Once each line had been traced it should be possible to

calculate the grid reference from the milepost mileage.

Stage 3 would create a spreadsheet to do this calculation. Stage 4 would draw a map showing the traced lines.

Should the master be HTML?

I was not satisfied that creating an HTML master, laid out by Internet Explorer and turned into a PDF by Acrobat Distiller was the right method to produce a book. The method seemed to give me complete control over how each page should look but was not without problems.

I found this method rather sensitive to the algorithms used by the browser and the distiller to lay out the HTML tables: a 'Windows software update' could change the Internet Explorer version so that the pagination went awry and caused me annoying difficulties with page layout.

I therefore decided to generate the page layout myself (i.e. generate a spreadsheet with the page, row and column number(s) as well as css style information and the relevant text for each cell but no HTML tags). I would then turn this into individual Draw files for each page of the book.

I had, by now, ended up with an HTML file containing nested tables – the outer tables specified odd and even page gutters for each page, plus a page number. The inner table contained a mixture of multi-line and multi-row cells. It was easier than it might seem to convert the HTML '<TR>' and '<TD>' tags to a spreadsheet row containing a row and column number.

Each row of the spreadsheet would have page number, first row, last row, left column, right column, style, font, text for each cell of text. This would identify how to justify the text, which font and size to use – this had already been worked out for the HTML output.

Stage 5 would therefore use a new method to produce individual Draw files for each page of the book. Stage 6 would turn these Draw files into PDF files in a 'hands off' batch process.

This had turned into six major pieces of work. I think the best way to explain this is to write an article for each stage.

Summarising

Summing up the process to produce the book, it starts by reading the detailed data from a spreadsheet into a BBC BASIC programme that produces another spreadsheet. This is then processed under RISC OS using BASIC into individual Draw files, each file forming a page of the book. The draw files are then batch processed into PDF files which are then combined into a single, print-ready PDF file using Adobe Acrobat.

This used my Iyonix (now retired), my ARMX6, an ARMiniX, Virtual RiscPC, Artworks, !MakeDraw as well as Adobe Acrobat 8.3 Standard under Windows 7.

Chris Hall *chris@svrsig.org*

The six stages of the project

The project started to look rather intimidating:

1: generate some map tiles for Ireland that showed old railways so that I could trace them and publish them;

2: trace each railway line, recording the grid reference at every point;

3: work out the grid reference of each signal box, knowing its distance along the line and then cross-check the result by plotting each signal box on a satellite view of the area;

4: draw railway lines onto a pastel-shaded background map;

5: convert spreadsheet cell data into Draw files;

6: convert several hundred Draw files to PDF.

Stage 1: Ireland and RiscOSM

Chris Hall

When I started to look at Ireland, I realised that no map tiles were readily available on-line but there was a detailed map available which showed old railway lines (at URL *http://www.railmaponline.com/UKIEMap.php*). This was, of course, copyright and so could not be republished. However I *could*, in principle, capture the necessary data to allow me to trace lines.

A screen view of these maps included a box showing the latitude and longitude at the centre of the map. If I displayed a map, did a screensave then just panned the map to show an overlapping view, and repeated this for the whole of Ireland, I would have the data I needed.

I therefore captured this at the largest scale that I could persuade myself to undertake. This required 700 1920x1080 screen shots, each showing the position of the centre of the view in latitude and longitude.

Because I had kept the browser window on the screen at the same position throughout, I could write a programme to read in each screen shot one by one and generate a draw file showing the filename and just that part of the screen shot that showed the location. I turned this into a PDF and did OCR to regenerate DATA statements of the form:

```
DATA filename, longitude,latitude
```

Another programme (Ire_Lg21) took each screen shot (sprite) in turn and placed it into a Draw file wrapper (with the screen shot(s) immediately above also included to avoid gaps, using the latitude and longitude of each to align them against each other). It then rendered the Draw file to screen using a transposition matrix to rotate it and a screen graphics viewport to crop it (after experimenting with the 'DrawFile_Render' SWI call's cropping matrix and finding it impossible to understand) so that each 1km x 1km grid square occupied 125 x 125 pixels on screen. Each grid square was then screensaved as a map tile.

It takes 103000 kilometre squares to cover Ireland and the programme ran for five hours overnight on my Iyonix to do this. I repeated the process for the more compact 3mkx3km and 6kmx6km squares that I also needed.

I now had a 125x125 pixel 24bpp sprite for each of the 103000 map tiles that I needed and 'just' needed to convert them into JPEGs. No problem I thought, I just need a batch converter. After a bit of searching, I wrote my own programme, in BBC BASIC, to do this (using the GDI+ library, see later). Overnight it processed the sprites and I now had the map tiles I needed.

I repeated the tiling process on an out-of-copyright map of Ireland – a sprite 6421x10573 pixels, some 259Mbytes in size which I turned into 103000 JPEGS each forming a 1kmx1km square.

Now I could proceed to stages 2 and 3 for Ireland (I had already done these stages for some parts of Britain). However the map tiles I had created to trace the lines in Ireland could not be used for any other purpose as they were copyright.

Using RiscOSM

I had purchased RiscOSM at the 2014 Wakefield show but had not started using it until I saw it again at the 2015 Wakefield show by which time it was able (for the first time) to display Irish national grid lines. My particular interest was to produce 125 pixel square jpeg map tiles

covering either a 1km or 3km square ortho-rectified to align with the National Grid, initially for Ireland (and its new ability to show grid lines would act as a check on my method).

The first step was to generate a map image where the area covered (latitude and longitude) was known. I chose an A0 landscape size sheet with a 1:50000 map and an A2 landscape size with a 1:100000 map (which both cover almost exactly the same area). RiscOSM imports Open Street Map data and allows you to export the map area as a vector graphic 'Draw' file. It also provides information about the latitude along the top and bottom edges and the longitude along the left and right hand edges (via the 'MapInfo' menu item).

The latitude and longitude used by RiscOSM is based on the WGS84/ ETRS89 spheroid model and is therefore the same as those used by GPS devices. Although the longitude value changes linearly from left to right on the map, the variation of latitude from top to bottom is more complex as the central position has a latitude that is not the average of top and bottom. This is necessary to have a constant vertical scale.

Conversion of the latitude and longitude to national grid coordinates is also complex. The first step is to transform to the equivalent latitude and longitude in the spheroid model that is used by the National Grid (this is the coordinate system used on OS maps to show the 5' graticules of latitude and longitude which therefore line up with the meridian at Greenwich, which GPS devices do not). Then another formula is used to tranform to easting and northing grid coordinates.

I now know the variation of longitude from left to right, have assumed that latitude is about linear from top to bottom and know how to transform to OSGB or OSI grid coordinates. I also have a vector graphic draw file which 'just' needs to be skewed and scaled to suit and then captured in 125x125 pixel squares.

There is an extremely useful OS

Something is wrong - the grid lines ruled by !RiscOSM (version 1.25) are too low (by about 100m at the centre of the map) - about 30m of this error is from drawing straight lines rather than a curve but there's something else as well that needs correction. By version 1.26 all was well.

routine called 'DrawFile_Render' which will transform and render a draw file to the screen using a transformation matrix where the screen position (x',y') is given in terms of the position (dx,dy) in draw units as follows:

```
x' = a.dx - c.dy + e      and
y' = b.dx + d.dy + f
```

For each km square the scale along the y-axis sets d, the scale along the x-axis sets a, the horizontal direction of the bottom edge sets b/d and the vertical direction of the left hand edge sets a/c. The screen position of the bottom left hand corner sets e and f. This gives the six parameters of the transformation matrix.

What happens to the top and right hand edges and the top right hand corner then, you might ask. The answer is that they will not be precisely rectangular and the error increases according to the number of grid squares you attempt to capture in one plot. I calculated the error at the top right as ~0.8 pixel with a 500x500 pixel plot (i.e. plotting 16 km squares at a time). Initially I was capturing 6x9 squares at a time to make full use of the screen size available - although this was faster it increased the error.

There was scope for error in these complex calculations and I introduced a debugging step where a drawfile was created covering exactly the same area as the map produced by RiscOSM but just having the corners of each 4km square screen plot marked, with their latitude, longitude and grid reference. I then imported both into Artworks with a SHIFT-drag and looked at how well the overlaid markers lined up with the grid line intersections produced by RiscOSM.

My assumption about latitude being linear produced errors around the mid height of the page. RiscOSM version 1.25 (1-May-2015) assumed that the horizontal grid lines were straight from the left hand edge to the right hand edge so there were errors at mid width. I had also used 3.14159 for pi rather than the BASIC function PI. Each of these produced an error of around 40m-100m whereas I was aiming at an error of about 1.2m (1 pixel).

It is interesting to note that the shortest distance between two points on a map based on latitude and longitude is not a straight line but rather a curve following the great circle through the two points. On an Ordnance Survey map, the grid lines each follow a great circle and so the shortest route is a straight line.

RiscOSM was revised (version 1.26) to plot grid lines more accurately so that horizontal grid lines will appear slightly curved. I have corrected my interpolation of latitude to use a formula to interpolate a function of latitude, ϕ namely $\log(\tan(\phi) + 1/\cos(\phi))$ and corrected 3.14159 to PI. The overlaid grid intersections now align to within a few metres. An error of ± 2.5m corresponds to ± 2 pixels or ± 0.00005° (longitude) or ± 0.00001° (latitude) and is the best that can be achieved where 'MapInfo' gives four decimal places of degrees.

In a couple of hours I managed to cover the whole of Ireland with 85 vector graphic images generated by RiscOSM using a longitude interval of 0.7° and a latitude interval of 0.3°. I noted down the latitude and longitude at the top, bottom, left, right and centre for each.

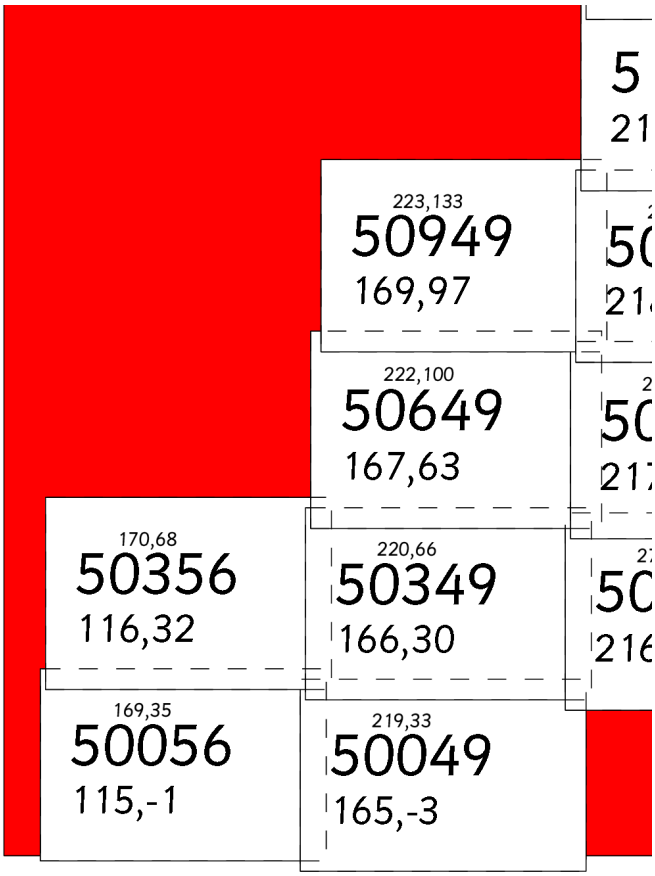
The first programme ('Ire_OSM20' is on the monthly disc) takes each of the 85 draw files in turn. It then projects each 4kmx4km square on the map onto the screen and captures the 16 km square map tiles, each as a sprite 125 pixels square. This takes about 8 hours on my ARMX6.

The programme 'SpritetoJPEG18.bbc' runs under BBC Basic for Windows and

batch converts the 138702 map tiles that were generated using 'screensave' as 62k sprite files into jpeg images of just a few kbytes each. This takes about 1½hours. The programme is more functional then polished but I've included it on the monthly disc anyway.

A 1:50000 map projected to cover 1km in 125 pixels (i.e. at a zoom level of 175%) looks about right for the most detailed tiles. I also needed coarser tilescovering 3km in 125 pixels and a 1:100000 map (zoom 117%) looked right for this. This time there were only 15022 tiles and it was much quicker.

The final product is a DVD-ROM produced for the Signalling Record Society titled 'Signal Box Register - volume 9 Ireland' which provides a map background of Ireland against which is projected the position of each signal box and a tracing of every railway line built in Ireland.



The SW corner of England showing overlap of maps, the filename '50356' indicates 50.3°N 5.6°W at the centre has grid coordinates (116.032) km at bottom left and (170,068) km at top right.

Using version 1.29 (beta) of !RiscOSM I was able, in a few days, to capture 206 A0 1:50000 and 206 A4 1:250000 vector graphic images covering Great Britain (see Artworks file Cov01) which I processed into 330876 1km squares, 36322 3km squares and 9239 6km squares, each 125x125 pixels. Version 1.29 adds an optional text title to the captured map making it unnecessary to type in the edge coordinates as they can now be read from the draw file - this speeded up the process somewhat.

The map tiles can be seen on-line at *svrsig.org/Map1940s.htm* with individual tiles generated by the Perl script *svrsig.org/cgi-bin/map1.cgi?x=147&y=030* where x and y are the grid coordinates of the map square in kilometres.

Overall I found RiscOSM to be an excellent piece of software and can thoroughly recommend it.

Chris Hall *chris@svrsig.org*

The finished map tile of the Penzance area.

Stage 2: Tracing Lines

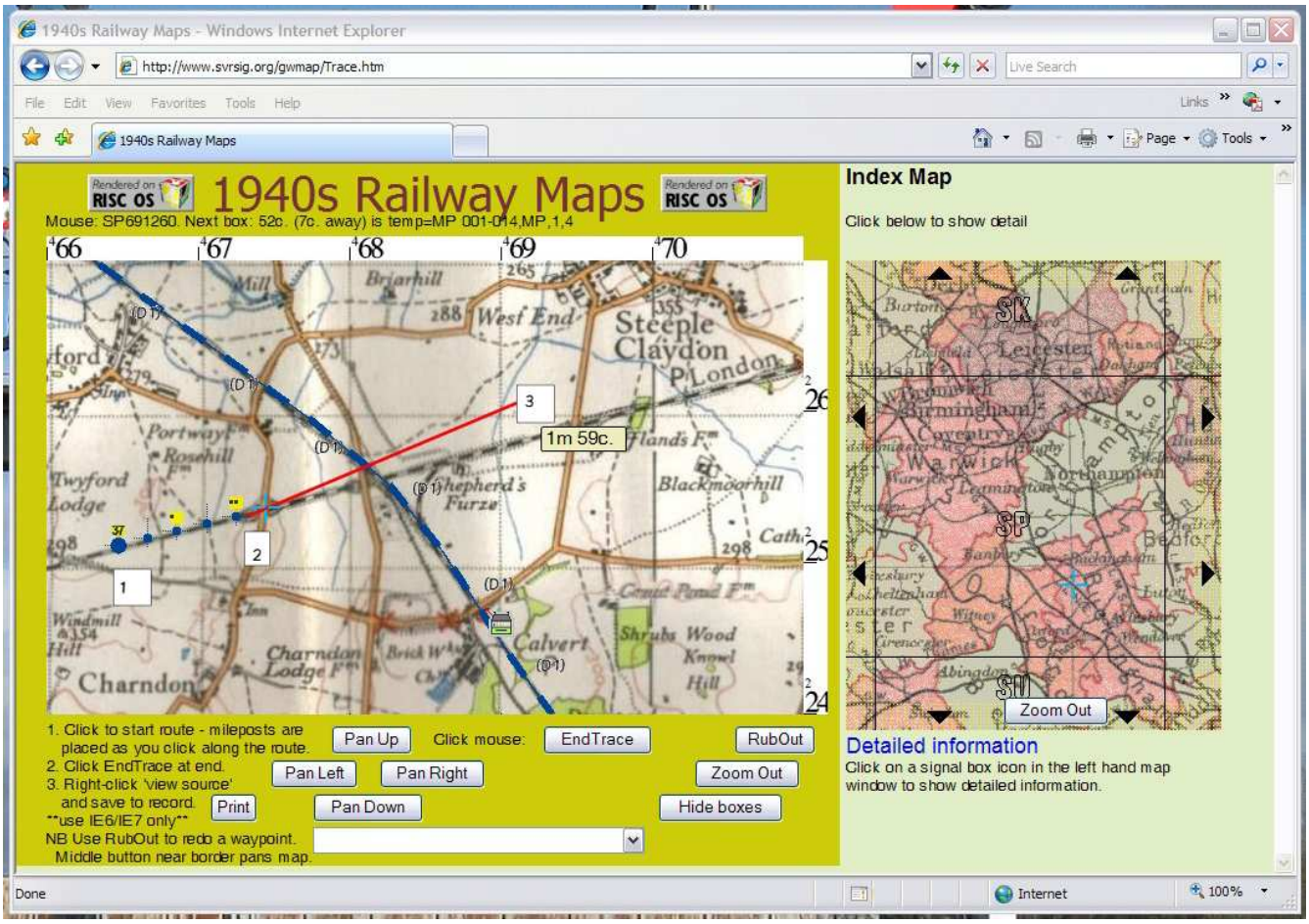
Chris Hall

I had established, by now, the grid references of many signal boxes using manual methods (measuring from reference points like tunnels on a printed OS map). I was therefore able to use some fairly simple Javascript to show the boxes overlaid on a 1940s OS map – see 'Superimposing on a map'. This manual method was, however, rather tedious.

An article in Computer Shopper in about 2007 described how to draw a vector graphic line in a browser and so I put together some javascript to track and record mouse clicks. The clicks would be over a set of map tiles, showing a 1940s 1" Ordnance Survey map (and therefore both open and closed railway lines), working out the distance travelled along the line and therefore the current milepost mileage.

A suitable set of map tiles for England, Wales and Scotland was readily available on-line as part of a project to capture grid references to match postcodes. As this would be a tool to gather information it would not need to be multi-browser capable: the vector graphics example code used a Microsoft library and the tracing has proved to work only in Internet Explorer 6/7 under Windows XP.

The mouse click positions are written into a browser window and the result captured (once tracing of the route is

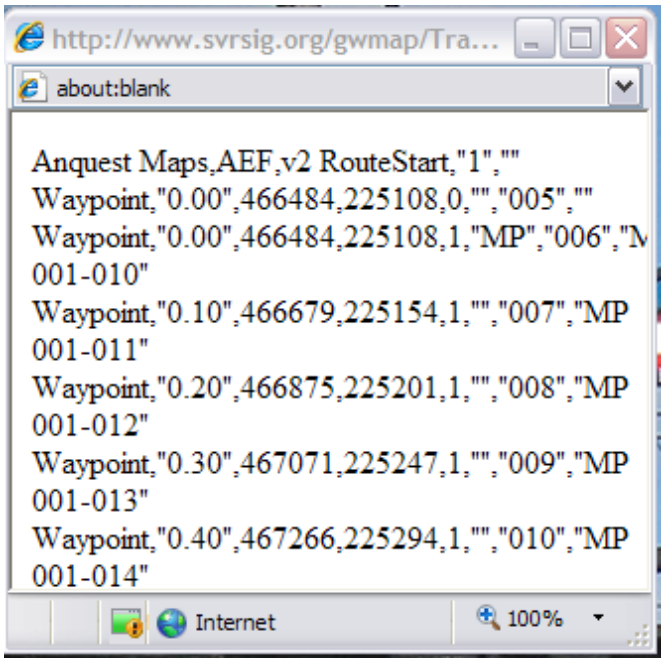


The tracing starts at the point marked '1', with a second click at '2' and the mouse being moved to the next point on the line ('3') with a red line showing the route to be traced between '2' and '3'. Blue dots show the route behind (the quarter (•) and half (••) mileposts can be seen). It requires Javascript to display correctly.

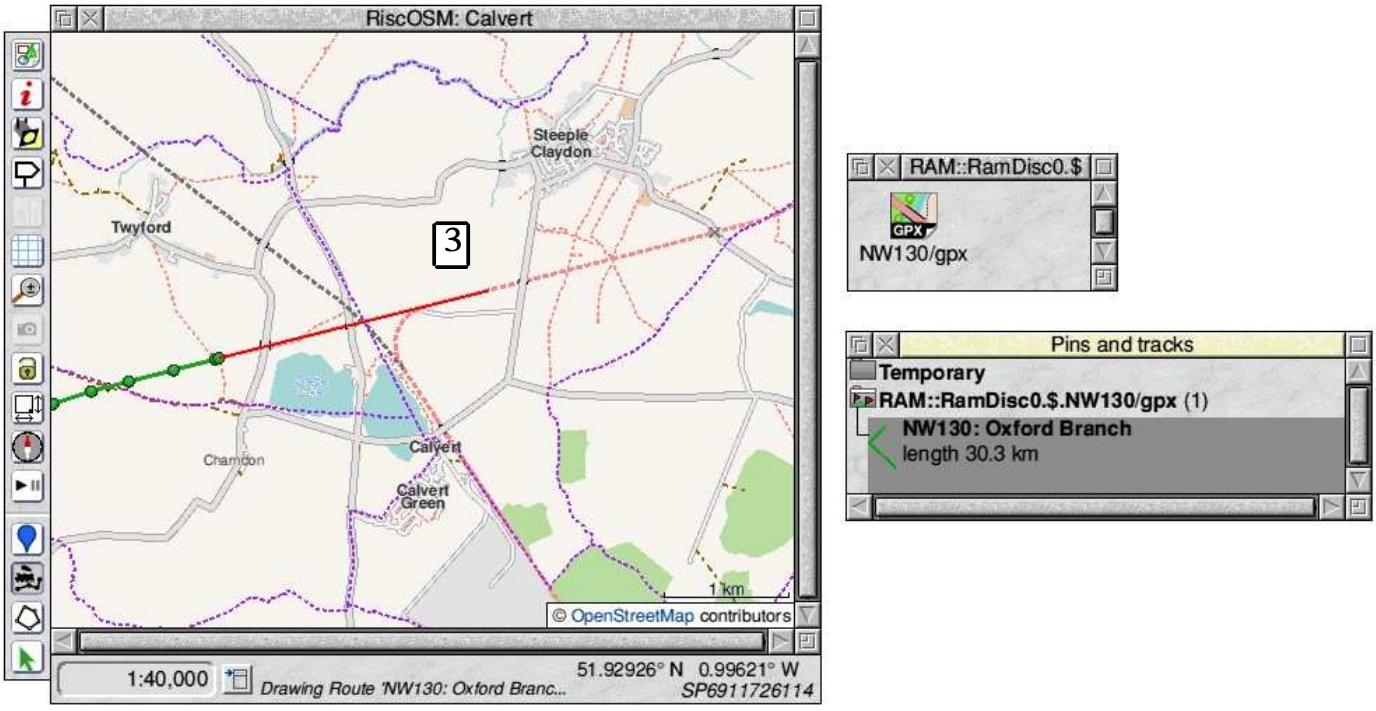
complete) by using 'view source' and 'save page'. The format of the text file is an Anquet 'export' format (.aef) which simply lists the grid references of waypoints along a route. This can be read directly into the Anquet mapping software and displayed and edited over a background of a modern 1:25000 or 1:50000 Ordnance Survey map. This page on my website (*http://www.svrsig.org/Trace.htm*) shows how this works. As the route is traced it is written into a separate window (see right) and the result is a file that looks like this:

```
Anquet Maps,AEF,v2
RouteStart,"1",""
Waypoint,"0.00",151452,32940,0,"","00-
5",""
Waypoint,"0.00",151452,32940,1,"MP","-
006","MP 001-010"
Waypoint,"0.10",151258,32995,1,"","00-
7","MP 001-011"
Waypoint,"0.20",151065,33050,1,"","00-
8","MP 001-012"   ...
```

It is a series of grid references of the mouse clicks, with 'milepost' markers dropped every ten chains so that you can see the



route you have just traced. Using Windows XP and Internet Explorer 6/7, the view shows a red line extending from the last click to the current mouse position indicating the line you are about to 'draw'. So far I have traced the whole of the Great Western, Midland, LNER (south of Doncaster), Southern, Scottish and London Underground railways.



A better method using !RiscOSM - an identical view, with the mouse at '3' but the steady red line follows the railway (shown as a red dotted line where still open and grey where closed) this time, not the mouse. The route already traced is shown in green with waypoints automatically generated to follow the railway.

Internet Explorer 7 vanishes

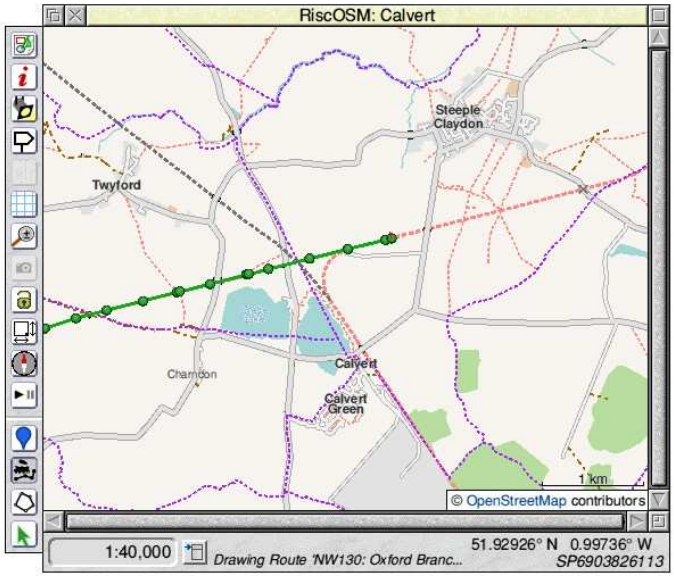
Suddenly I found that Windows XP and Internet Explorer 6 were disappearing. At first I built an XP compatibility box on my Windows PC but I upgraded to Windows 7 and it seemed sensible to find a different method for tracing lines.

RiscOSM

RiscOSM provided the solution - it can draw tracks from the last point to the mouse (or the nearest point to it) constrained to follow a footpath or railway or road etc. making the job of tracing a line much easier. Some abandoned lines are discontinuous across a removed bridge but that section can be filled in by forcing a 'jump' across the gap using the CTRL key.

RiscOSM allows the route to be exported in GPX format (i.e. using latitude and longitude) and !MultiTask allows such an exported GPX route to be converted to Anquet AEF format.

Chris Hall *chris@svrsig.org*



After a single mouse click, RiscOSM has extended the route along the railway with extra waypoints so that it follow the curves. Before these had to be done manually.

Stage 3: Grid references

Armed with the grid reference of each signal box and the route followed by each railway line, I knew that I would be able to correlate milepost mileage with grid reference. Another 'tweak' to the growing BASIC programme would produce a spreadsheet full of formulae that could convert milepost mileage to grid reference. No more manual methods!

The screenshot below shows the traced line (actually from Welwyn where it left the main line), the milepost mileage and the calculated grid reference. This allowed me to establish grid references for each signal box.

These grid references needed to be checked. Now I had obtained the tracing of each line and worked out where each box was located, I could show both lines and boxes on a map background. URL *http://www.svrsig.org/gwmap/Map1940s.htm* shows this (it uses Javascript extensively and many RISC OS browsers (including Netsurf) cannot handle this.

The only features that were really definitive on the map, that would confirm that the method worked, were level crossings.

An article in Computer Shopper in about 2006/7 showed how to display 'pushpins' – a small graphic icon with pop-up text – overlaid on a browser window displaying an aerial view from Google Earth. This would therefore be able to show an aerial view, defined by latitude and longitude, with icons showing the position of signal boxes, defined by an OSGB grid reference.

A few lines of javascript later, I had added a 'satellite view' button but was disturbed to find that, for example, the location of Penzance signal box (which is still open and can be seen clearly on a satellite view) looked correct on the Ordnance Survey map (so its grid reference was right) but it was misplaced on the satellite view.

Time for some research! The

H	I	J	K	L	M	N	O	P
Railref	Box name		Drg no.	Xgrid	Ygrid	MPM	Calculated grid ref	
							MPM	Grid Ref
GN 030-240	Hatfield No.2			523200	208700	18.04	18m.4c.	TL 232 094
GN 032-020	Ayot		S465	522100	214400	22.12	22m.12c.	TL 221 144
GN 032-030	Wheathampstead			517600	214300	25.11	25m.11c.	TL 176 143
GN 032-040	Harpenden			514400	215100	27.22	27m.22c.	TL 144 151
GN 032-050	Luton Hoo			511900	218000	29.57	29m.57c.	TL 119 180
GN 032-060	Luton East		S62	509300	221400	32.43	32m.43c.	TL 093 214
GN 032-070	Luton West			508900	221500	32.61	32m.61c.	TL 089 215
GN 032-075	Luton Station			0	0		nk	nk
GN 032-080	Luton West			508100	221600	33.21	33m.21c.	TL 081 216
GN 032-090	Chaul End		S124	506100	222100	34.45	34m.45c.	TL 061 221
GN 032-100	Dunstable Church St			502500	222000	36.75	36m.75c.	TL 025 220
GN 032-110	Dunstable East			501000	222500	37.78	37m.78c.	TL 010 225

The highlighted cells are manual inputs - the milepost mileage where the tracing starts (18m4c) and ends (37m78c). Column N is the milepost mileage of each location and the corresponding calculated grid reference is in column P.

Ordnance Survey grid reference is based on the OSGB36 (1936) coordinate system and although I had used the recommended (and highly complex) formulae to derive a latitude and longitude from the grid reference, this was also based on the OSGB36 coordinate system. I discovered that the latitude and longitude used by Google Earth (and satellite navigation systems) is based on the more modern WGS84 (1984) system. Although the error is 'only' a few hundred yards (more in Scotland), that looks quite big when the satellite view can be expanded so that a few hundred yards fills more than the whole computer screen!

GPS systems are worldwide: local coordinate systems define the angular momentum of 'local' tectonic plates to be treated as zero. Conversion between the two requires another complex formula and Penzance box suddenly appeared in the right place.

Chris Hall *chris@svrsig.org*

Stage 4: Drawing Lines on a Map

Chris Hall

I now have a list of grid references defining the route of each railway line, a list of signal boxes, each with a grid reference, and a scan of an out-of-copyright map of England which is 4588x5836 pixels, some 100Mbytes in size.

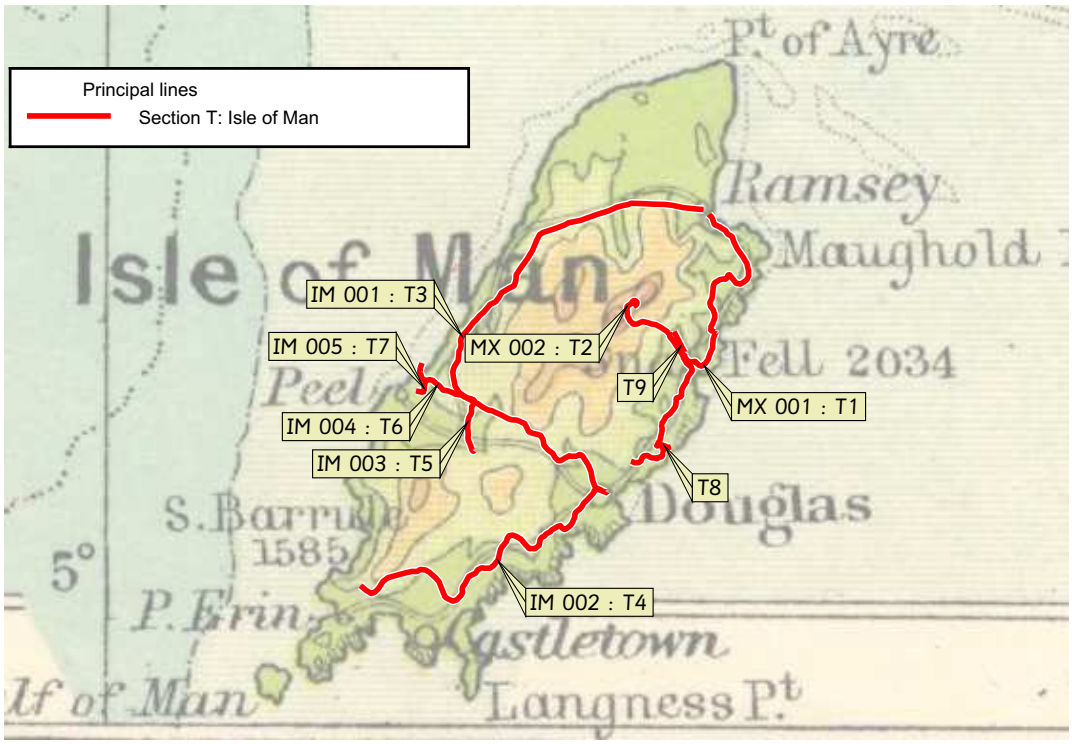
I therefore wrote a programme (MaploM46) in BASIC to look at a list of lines along with their traced route, which defined a bounding box. It then worked out a suitable part of the scanned map to cover the right area. It then reads this into a user sprite area, washes out the colours to pastel shades and surrounds it with the necessary draw file headers. Each line was then plotted as a vector graphic the result saved as a Draw file. Each Draw file forms a page in the book.

Because the sprite was a scan of a map, the gird references at each corner had to be estimated and the line drawing skewed and scaled to fit. Old maps didn't bother to show the route of railway lines (or anything else) very precisely so this was essentially a compromise.

Reading the resulting draw file into Artworks, generates a PDF which includes an RGB JPEG with CMYK process colours given to the vector lines (done by ArtWorks on draw import). The PDF is suitable for commercial printers (they tolerate RGB JPEGS but complain about RGB process colours).

I now had the coloured map pages for the book.

Chris Hall *chris@svrsig.org*



List of lines in Section T: Isle of Man

T1	MX 001	Douglas to Ramsey
T2	MX 002	Laxey to Summit
T3	IM 001	Douglas to Ramsey
T4	IM 002	Douglas to Port Erin
T5	IM 003	Foxdale to Kirk Michael
T6	IM 004	St Johns to Peel
T7	IM 005	Knockaloe Branch
T8	------	Lhen Coan to Sea Lion Rocks
T9	------	Laxey to Adit

This image appears in the book to introduce Section T - the lines on the Isle of Man.

Stage 5: Draw files - hundreds

Chris Hall

I now realised that converting a spreadsheet of row/ column/ text entries was a bit daunting. I was, pretty much, having to do all the text formatting and justification that a browser would do. The programme (ReadReg089) did actually work - a mere 1542 lines of BASIC.

However I did need to create a RISC OS Arial font, using the excellent !effTTT TrueType font Translator from the Electronic Font Foundry and use !FontEd to add some special characters (eighths fractions and feet and inches marks).

The row/ column/ text spreadsheet is shown in the screenshot below - this is not designed for reading or editing, just for processing. It may have hundreds of thousands of rows (and Excel 97 won't even load it).

The corresponding extract from the spreadsheet and the relevant page (page 188) in the book (Volume 9 Signal Box Register Ireland) are shown below.

Chris Hall *chris@svrsig.org*

	A	B	C	D	E	F	G	
25192	188	1	1	1	14	phead	phead	Section 12 - K: Great Southern & Western Railway
25193	188	2	2	1	14	none line	subhead	K15: Clara & Banagher Junction to Banagher
25194	188	3	3	1	14	lhead	150	Railway: construction begun by Midland Counties & Shannon Railway Co. but on their failure complete
25195	188	4	4	1	1	aa	std	
25196	188	4	4	2	2	ab	std	
25197	188	4	4	3	3	ac	std	
25198	188	4	4	4	4	ad	std	
25199	188	4	4	5	8	aad	std	Signal Box
25200	188	4	4	9	12	aeh	std	Locking Frame
25201	188	4	4	13	13	am	std	
25202	188	4	4	14	14	cn	std	
25203	188	5	5	1	1	da	std	Ref
25204	188	5	5	2	2	db	std	M
25205	188	5	5	3	3	dc	std	Name
25206	188	5	5	4	4	dd	std	#
25207	188	5	5	5	5	ae	std	Opened
25208	188	5	5	6	6	af	std	Closed
25209	188	5	5	7	7	cg	std	Type
25210	188	5	5	8	8	ah	std	Construction/Size
25211	188	5	5	9	9	ci	std	Type
25212	188	5	5	10	10	cj	std	Cen's
25213	188	5	5	11	11	ck	std	Size
25214	188	5	5	12	12	al	std	Date
25215	188	5	5	13	13	dm	std	BS
25216	188	5	5	14	14	en	std	Notes
25217	188	6	6	1	14	solid none none	italic	from Clara & Banagher Junction (see GX 032-030, Section K14) - milepost mileage 0.00
25218	188	7	8	1	1	aa	std	GX 033-010

The rather incomprehensible spreadsheet defining where on the page each cell should be placed.

Section 12 - K: Great Southern & Western Railway

K15: Clara & Banagher Junction to Banagher

Railway: construction begun by Midland Counties & Shannon Railway Co. but on their failure completed by Clara & Banagher Railway Co., worked by GSW from opening and absorbed by them w.e.f. 14.05.1895
Line: opened 29.05.1884; closed 24.02.1947 (passengers), 31.12.1962 (goods)
Track: 5' 3" (1600mm) gauge; single
Electrification: Nil
Block: Clara & Banagher Jcn-Ferbane: 1884-1893: TS&T with AB; 1893-1920s: ETS (large); 1920s-14.10.1958: ETS (miniature); 14.10.1958-1962: OES+TS
Ferbane-Banagher: 1884-1893: TS&T with AB; 1893-14.10.1958: ETS (large); 14.10.1958-1962: OES+TS
Mileage: from Clara & Banagher Junction

				Signal Box				Locking Frame					
Ref	M	Name	#	Opened	Closed	Type	Construction/Size	Type	Cen's	Size	Date	BS	Notes
from *Clara & Banagher Junction (see GX 032-030, Section K14) - milepost mileage 0.00*													
GX 033-010	10.16	Ferbane		00.00.1884	00.10.1958	RS(GSW) BTW		GF		9			K121
								E=		10	by 1961		

Part of the same page from the finished book.

Stage 6: Producing PDFs

Chris Hall

I have tried several methods of producing PDFs to be sent to commercial printers and have settled on one method which seems to work best. I had two principal source documents - one was a multi-page book generated from a set of data using a BBCBASIC for Windows programme and some source spreadsheets and one was a single page coloured map, produced using !MakeDraw. The method for each was different and I'll describe them separately.

Single page coloured map

The coloured map itself is produced as a draw file from BASIC from a scanned out-of-copyright map of England, Wales and Scotland in the early 20th Century, Anquet 'AEF' route files of every railway line in the book and a note of which lines to show on the page. I produced a PDF from the draw file by using !ArtWorks to 'Export/PDF' setting 'include fonts' to ensure they were embedded. Producing a single PDF from a single draw file was reliable and proved satisfactory for printers.

This was because all the vector graphic lines in the draw file were automatically converted to CMYK process colours on import to ArtWorks (a bit colour saturated as K was wrongly set to zero). Although the embedded jpegs were RGB, this was apparently tolerated by most printers (one supplier has a fussier printer and runs an expensive version of Acrobat to convert the RGB jpeg images to CMYK for me).

Multi-page book

Initially (in 2006) I produced the data in the form of an HTML master, created using BBC BASIC, containing nested tables with css formatting to specify page breaks and cell borders. Producing a PDF was simply (!) a case of loading the master into IE6 with very specific page borders and printing to Adobe Acrobat Standard 8.3 with options to embed text fonts and to use 'high resolution'. This got more and more complex as I was having to second guess the browser about some of its formatting and, particularly, how much I could get on one page before adding page number and forcing a page break (using css).

At this stage I could include any characters so long as I made sure to convert any top-bit-set characters to their HTML equivalent. I had to work out how Excel 97 stored these characters in csv files saved from it (various things like eighths fractions, feet and inches marks, sexed quotes etc.). However any 'windows update' that updated IE6 might change the formatting subtly and affect my page breaks. Also I found that if you had too many (more tham 100 or 200) jpeg or gif images on the HTML master then not all would necessarily be rendered. Fine on screen but useless for a book.

When IE8 was produced it was so buggy that it wouldn't even load my 450 page HTML document (7.3Mbytes) at all, just crashing. It had 16402 internal HTML links and 990 nested tables, each having individual css-defined borders. I had to split the document into two or more parts to make sure that all pictures would get rendered.

Having already produced a tabular document, using HTML markup for the cells and css formatting for whether it had ruleoffs above, left and right of the cell and what font was required, in a BBCBASIC programme, I decided that it would be relatively simple to produce a spreadsheet

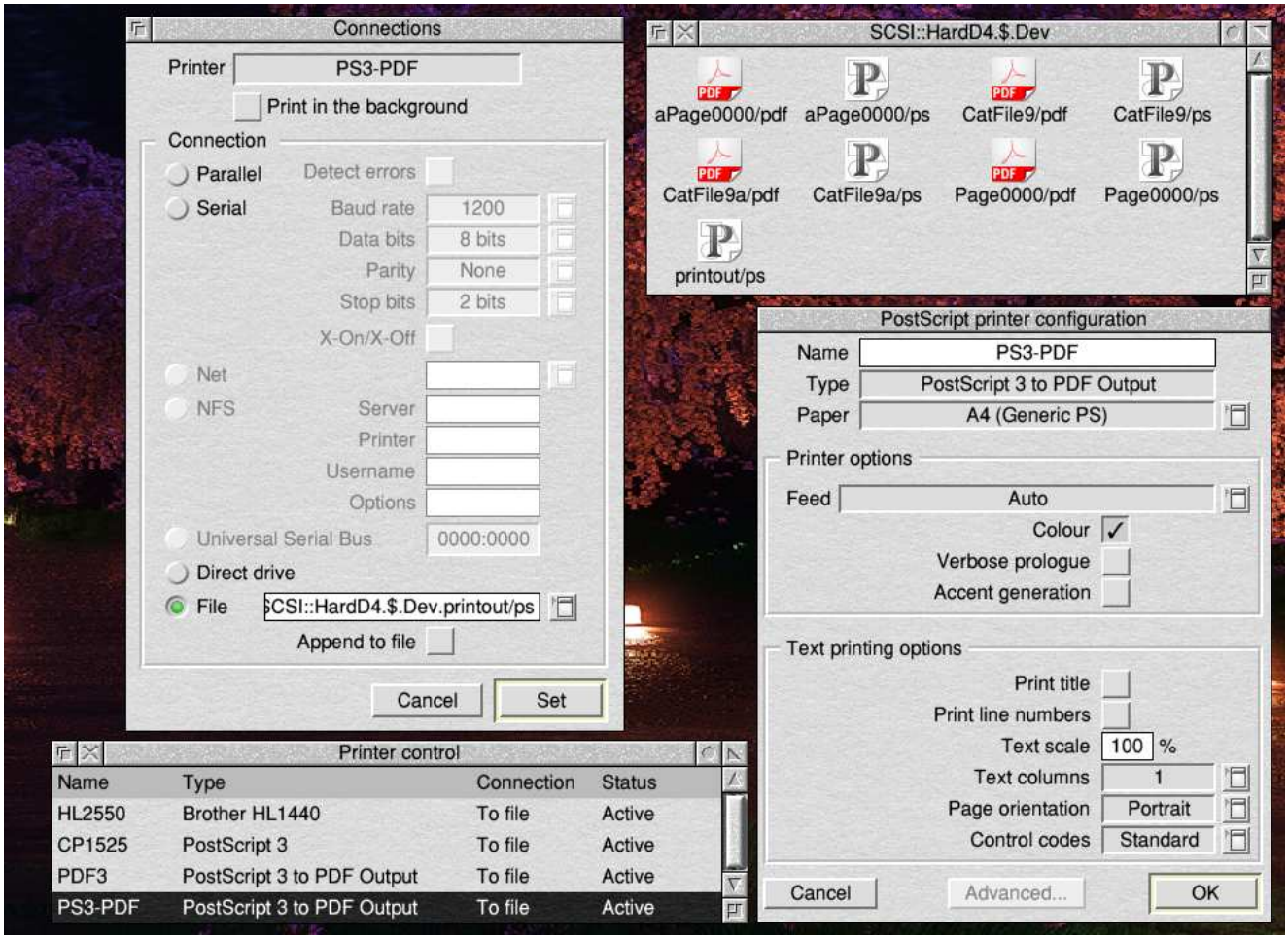
instead, each line identifying page number, leftmost cell cloumn, rightmost cell column, highest cell row and lowest cell row followed by css format type for the cell, css format type for the text and then the text itself. The result was a csv spreadsheet which had 123230 lines of such entries.

All I had to do was to write a programme that would produce an individual draw file for each page, justifying the text within the cell it occupied, spacing adjacent cells appropriately, include jpeg pictures at a specified dpi (on the finished page) or size. I therefore created an Acorn 'Arial' font (using the Windows font 'Arial' and the !EFFTTT (true type translator) to create the font and !FontEd to add eighths fractions and feet and inch marks) and wrote the programme. Now the two forks have converged.

Hundreds of draw files

Now I have 450 draw files, all produced using BBC BASIC, which form the book. I need a new method of producing PDFs because, even though Martin Wuerthner produced a trial version of ArtWorks for me (the South West show was rather quiet that year and he did this during the show!) that would allow me to drag all 450 draw files onto ArtWorks and place each one on a new page automatically (provided I held down the CTRL key throughout - sellotape helped here a lot) it took so long (and filled memory after about 12 coloured pages, some of which were about 40Mbytes) that it became too difficult.

I have the Postscript 3 printer drivers which can produce a postscript file if you print something under RISC OS. First set up the driver to produce the postscript as a file:

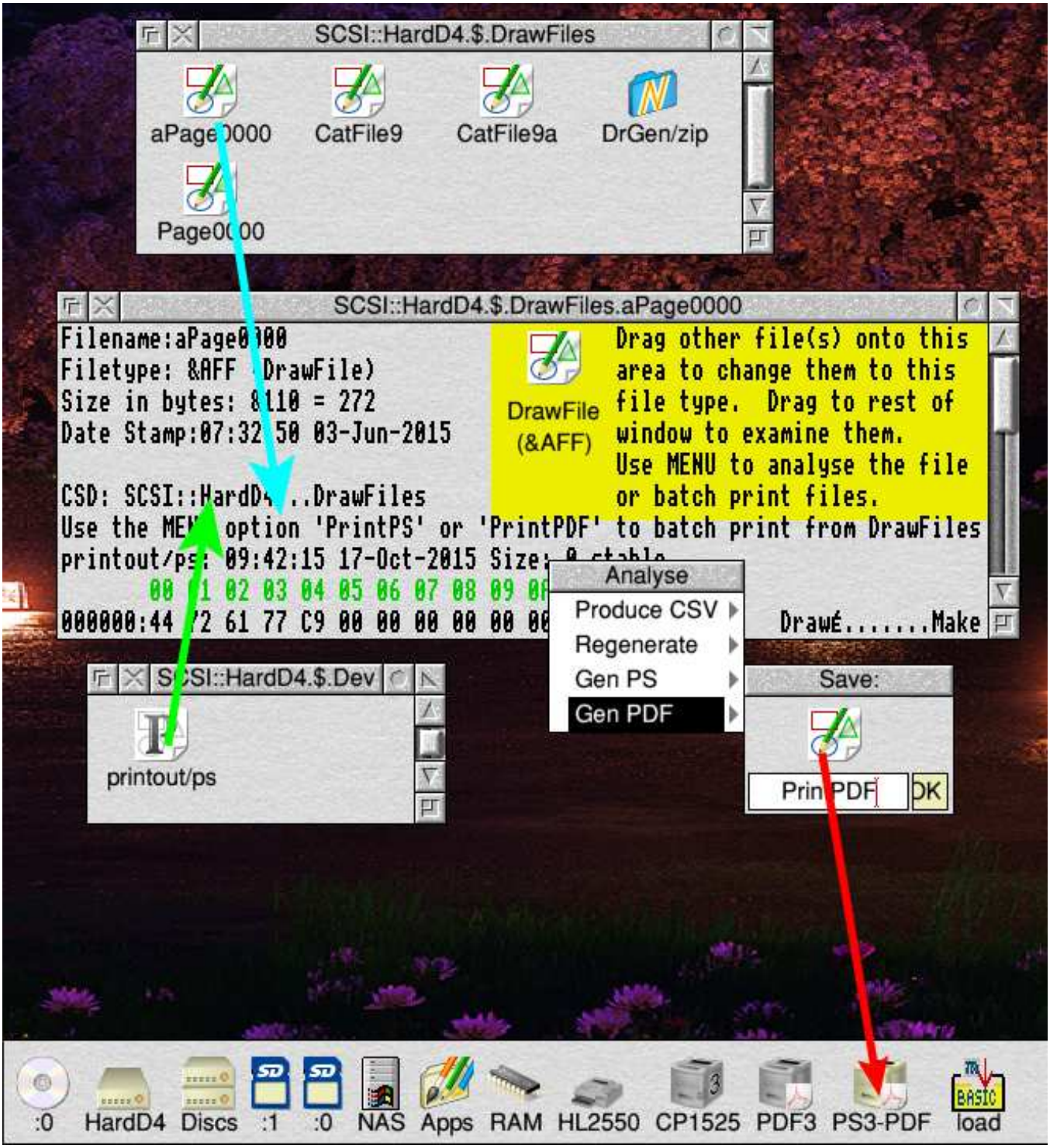


Drag a draw file onto the postscript printer driver icon and a postscript file 'postscript/ps' is produced wherever you specify. The (slightly obscure) command inside a BASIC programme:

```
SYS "Wimp_StartTask","ps2pdf13
"+FNUnix(LEFT$(pout$,LEN(pout$)-
11))+AwaitFile$+".ps
"+FNUnix(LEFT$(pout$,LEN(pout$)-
11))+AwaitFile$+".pdf"
```

will convert this file to a PDF 'postscript/pdf'. All I need is something that will do that for a directory full of draw files, dragging each in turn to the icon, waiting for the 'postscript/ps' file to be created, rename it to the leafname of the last dragged file plus '/ps', convert it to a pdf with same leafname and then repeat until all files have been dragged.

What does this look like?



With the application !MultiTask running (the yellow icon on the icon bar), a click on its icon bar icon will open the window displaying file contents.. First drag the 'printout/ps' file icon onto it (so that it knows where to look for each postscript file) as shown by the green arrow. Then drag a draw file from another directory which contains many other draw files (cyan arrow). Then click MENU over the window and drag the 'PrintPDF' file icon onto the postscript printer icon (red arrow). The filer window 'Dev' will fairly quickly fill up with 'pdf' and 'ps' files created from each draw file and you will have converted all the draw files to PDFs. Use Adobe Acrobat Standard to make a single multi-page PDF from the individual page PDFs.

If you want to convert an Impression file into a PDF, then simply select the print driver and print from Impression - this creates a file 'postscript/ps'. Then run a short BASIC programme in a task window (!MultiTask will automatically set the USD to the directory containing the BASIC programme and the 'postscript.ps' file just produced) which contains:

```
SYS "Wimp_StartTask","ps2pdf13
printout.ps printout.pdf"
```

and the printed output from Impression will become a multipage PDF.

What problems did I encounter?

The special characters (such as sexed quotes, fractions, feet and inches marks etc.), stored in Excel 97 or Word 97 as top-bit-set characters, had been translated in stage 5 and so appeared correctly in the final PDF. Some images, however, caused me a lot of difficulty.

RISC OS 5 uses 'new format' sprites containing low colour depth (8bpp and lower) extensively but Publisher cannot print them (but displays them correctly on screen). ArtWorks refuses to load them and even if you sneak them in inside a Draw file, will not display them. Strangely though, it prints them correctly. However it will not export them to PDF, giving an error.

I nearly tried using text area objects but quickly discovered that despite being part of the Draw specification since before 1992, neither ArtWorks nor Publisher could display them, ArtWorks would error and show garbage text.

Only in the latest update for Impression-X (so I am told) can it accept JPEG objects embedded in a Draw file (or dragged onto a window) - these were added to the Draw specification with RISC OS 3.60 in 1995.

I have therefore produced a Draw file (DrawEx) with all these difficult objects as an example, see below.

Chris Hall *chris@svrsig.org*