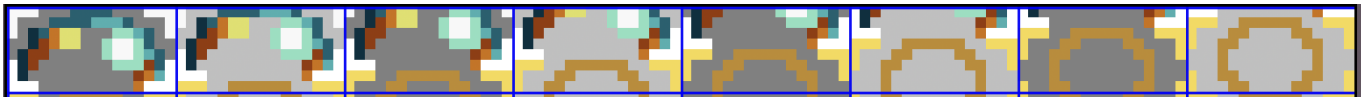
Vertical scrolling can be achieved using RRB also using the y offset and rowmask featues, in order to do this you need to have 2 RRB layers per line, one for the current line of map and one for the next line of map that is super imposed on top and masked.

A convention that I use is that and increasing YScroll value will scroll the map data up the screen.

Ideally, map data is laid out in the following format

map line 0	**A**	**B**	**C**	**D**	**...**
map line 1	E	F	G	H	...

For each screen row you will want to draw map line (yscroll>>3) AND map line ((yscroll>>3)+1).



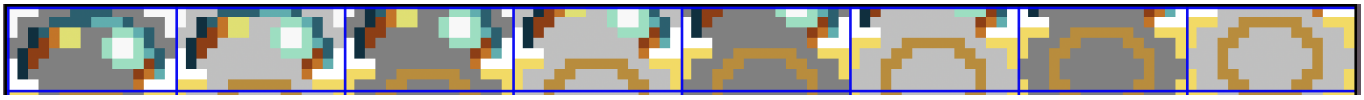
Using (YScroll & 7), we can simply use FCM charatcer data Y offset (screenRam byte1: bits 5-7) to shift up the character data and mask accordingly using the following table.

	line 0			**line 1**		
(yscroll & 7)	mapline0	+ve y offset	rowmask	mapline1	+ve y offset	rowmask
0	chr	0	%11111111	chr-1	0	%0000000
1	chr	1	%01111111	chr-1	1	%10000000
2	chr	2	%00111111	chr-1	2	%11000000
3	chr	3	%00011111	chr-1	3	%11100000
4	chr	4	%00001111	chr-1	4	%11110000
5	chr	5	%00000111	chr-1	5	%11111000
6	chr	6	%00000011	chr-1	6	%11111100
7	chr	7	%00000001	chr-1	7	%11111110

And here we run into the issue, you can see that to correctly overlay map line1 using +ve character Y offset you need to subtract 1 from the character index in order to shift up that map line's character data into the current row. The only way I found of achieving this was to have two copies of the map in data, one that is the true map data and one copy that is map data char index - 1.

(Future proposal - allowing -ve FCM character data Y offset)

As with RRB sprites, the feature add of -ve character data Y offset would solve this issue.



You could than use the table as follows (map line 0 is being shifted up and map line 1 is being shifted down)

	line 0			**line 1**		
(yscroll & 7)	mapline0	+ve y offset	rowmask	mapline1	-ve y offset	rowmask
0	chr	0	%11111111	chr	0	%00000000
1	chr	1	%01111111	chr	7	%10000000
2	chr	2	%00111111	chr	6	%11000000
3	chr	3	%00011111	chr	5	%11100000
4	chr	4	%00001111	chr	4	%11110000
5	chr	5	%00000111	chr	3	%11111000
6	chr	6	%00000011	chr	2	%11111100
7	chr	7	%00000001	chr	1	%11111110

Now you would only need one copy of the map and the accessing of mapline1 data doesn't need to be modified.