detector character is to be moved.

```
3000DEF PROCtest_keyboard
3010 REM ** UP ? **
3020IF INKEY(-58)=-1 THEN PROCmove(0,-1)
3030REM ** DOWN ? **
3040IF INKEY(-42)=-1 THEN PROCmove(0,1)
3050REM ** RIGHT ? **
3060IF INKEY(-122)=-1 THEN PROCmove(1,0)
3070REM ** LEFT ? **
3080IF INKEY(-26)=-1 THEN PROCmove(-1,0)
3090ENDPROC
```

THE PROCEDURE 'MOVE'

This procedure is central to the program. Within it the detector and assistant characters are moved, and collisions with mines are tested for. Let us first look at the section of the procedure that controls the movement of the characters.

Two parameters are passed into 'move' from the procedure 'test-keyboard'. These are accepted into the variables delta-x and delta-y for use within 'move', and correspond to the change to be made to the x and y co-ordinates of the mine detector. For example, if the cursor-up key were pressed then the values 0 and −1 would be passed to 'move'. The instructions: xdet = xdet+delta-x and ydet = ydet+delta-y cause the co-ordinates of the detector to be updated. In the case of cursor-up, 0 is added to xdet and −1 is added to ydet, effectively subtracting one from its value. This seems to imply a move of one unit *down* the screen, but we must remember that the origin for character positions is the top left corner and y values increase down the screen. Thus, reducing ydet by one causes an upward movement of one character cell. You may wish to verify that the values passed for the other three directions do in fact correspond to the correct alterations of xdet and ydet. It would be quite feasible to use this system to include diagonal movements. Passing the values (1,-1) to 'move' would cause the detector to move diagonally one cell up and one cell right. However, other keys would have to be introduced at this stage to allow diagonal control from the keyboard.

Here is the listing for the 'move' procedure:

```
3220 DEF PROCmove(delta_x,delta_y)
3230REM ** RUB OUT OLD POSITIONS **
3240COLOUR 1
3250PRINTTAB(xdet,ydet);" "
3260PRINTTAB(xman,yman);" "
3270REM ** MOVE DETECTOR **
3280xdet=xdet+delta_x
3290ydet=ydet+delta_y
3300REM ** TEST FOR LIMITS **
3310IF xdet>17 THEN xdet=17
3320IF ydet>25 THEN ydet=25
3330IF xdet<2  THEN xdet=2
3340IF ydet<1  THEN xdet=1
3350REM ** CALCULATE MAN'S COORDS **
3360xman=19-xdet
3370yman=26-ydet
3380PROCconvert(xman,yman)
3390IF POINT(xgraph,ygraph)=2 THEN PROCexplode(xgraph,ygraph)
3400PROCconvert(xdet,ydet)
3410IF POINT(xgraph,ygraph)=2 THEN PROCfound_mine
3420PROCposition_chars
3430ENDPROC
```

Before the x and y co-ordinates of the detector are altered we must first erase the old positions of the detector and the assistant. Lines 3250 and 3260 use the old values of xdet, ydet, xman, and yman to PRINT spaces over the old characters. As the new characters will be PRINTed in red (logical colour 1) the colour command is used in line 3240 to set the current foreground colour to 1. Lines 3280 and 3290 update the co-ordinates of the detector as described previously. Before actually PRINTing the detector in its new position, tests must be made to ensure that we are not incrementing or decrementing co-ordinates outside the area we have defined as our minefield. The upper and lower limits of xdet and ydet are tested in lines 3310 to 3340. Here it has been decided that if the detector reaches a boundary then it will stay there until moved in the opposite direction. For example, line 3310 tests to see if the right-hand edge of the minefield has been reached, denoted by an x co-ordinate of 17. If an attempt is made to increase xdet past 17, then this line simply resets the value to 17. It would have been equally possible to create a 'wrap-around' effect in which the detector, on reaching the right-hand boundary, would next appear back on the left-hand side of the screen. To produce a wrap-around effect at the right-hand edge of the minefield, we alter line 3310 to read:

```
3310 IF xdet>17 THEN xdet=2
```

You may wish to alter this and the other three boundary tests to provide wrap-around on all edges of the minefield.

One of the rules of our game is that while the player moves the mine detector around the minefield destroying mines the player's assistant mirrors every move. In order to do this, we must automatically update the assistant's co-ordinates, which are related to the detector's co-ordinates by a simple pair of formulae as shown in lines 3360 and 3370. To demonstrate how these produce mirror movements let us look at the relationship between the x co-ordinates (xman=19−xdet).

Initially, xdet is 2 and xman is 17. If the detector moves one place to the right, xdet will increase to 3. Using the formula above, xman will be calculated as 19−3 = 16. This means that the assistant moves one place to the left. If xdet moves right again, then xdet will become 4 and xman will be 15, and so on. The y co-ordinates operate similarly.

Before we PRINT the detector and the assistant we have one remaining task. We must check to see if either the assistant or detector is moving into a character cell that is already occupied by a mine.

Mapping

The game mixes high resolution graphics with the BBC/Electron text display. This has its advantages but means two different co-ordinate systems must be mixed, one for graphics and one for text. The BBC/Electron has several different text formats and so each has its own co-ordinate system. The game uses mode five, which gives 20 characters across the screen and 32 down. This is shown in the top and left-hand part of the diagram.

The machines also have three different graphics resolutions, but at least these all use the same co-ordinate system. This treats all modes as having a resolution of 1,280 by 1,024. This is shown in the bottom and right-hand part of the diagram.

The program uses the high resolution co-ordinate system to read points off the low resolution text display. This means converting a character position to a high resolution co-ordinate. To do this the horizontal co-ordinate (XCHAR in the program) must be multiplied by 64 and the vertical co-ordinate (YCHAR) by 32. One other problem has to be overcome. The text screen co-ordinates start with zero at the top of the screen and count down, while the graphics co-ordinates start with zero at the bottom and count up. This is easily solved by subtracting 32*YCHAR from 1,023