BBC BASIC allows us to test any point on the screen to see if that point is lit in a certain colour. POINT(X,Y) will return the colour of the pixel at position (X,Y). We can use this to see if the colour of the cell we are about to move into is green (i.e. it contains a mine). There is only one snag. POINT(X,Y) uses the high-resolution co-ordinate system to specify the point to be looked at. If we want to use this command for our game we must first convert our character cell co-ordinates into grahics co-ordinates. The easiest point in the cell to specify would be the centre.

In the last section we worked out that in mode 5 each character cell is 64 graphics units wide and 32 graphics units high (see page 404). Multiplying xchar by 64 will give the xgraph co-ordinate of the edge of the cell in question. Adding a further 32 to xgraph will give the x co-ordinate of the centre of the cell. Calculation of ygraph is rather more complex as the two systems run in opposite directions. At the top of the screen ygraph is 1023. Working down, 32*ychar would bring us to the top of the specified cell; moving down a further 16 units would bring us to the y co-ordinate of the centre of the cell. The following procedure can, therefore, be designed to convert character co-ordinates to graphics co-ordinates:

```
3720 DEF PROCconvert(xchar,ychar)
3730xgraph=64*xchar+32
3740ygraph=1023-(32*ychar+16)
3750ENDPROC
```

We can see the true value of being able to pass parameters between procedures if we look back to the procedure 'move'. The 'convert' procedure is used twice: first of all, to convert the co-ordinates of the assistant into graphics co-ordinates, and these values are then used in line 3390 to test for the colour green. (Remember that although the assistant's co-ordinates have been updated, the character has not yet been PRINTed in its new position.) If the colour present is green then the program jumps to another procedure to display an explosion. The 'convert' procedure is used for the second time in line 3400, but in this instance the character co-ordinates of the detector are calculated. Line 3410 then tests to see if the cell is occupied by a mine. If it is, the procedure 'found-mine' is called. Finally, the detector and the assistant are PRINTed in their new positions by calling 'position-chars'.

We shall be looking at the procedure 'explode' in the next instalment, and so for now let us put a dummy procedure in its place. Type in the following lines:

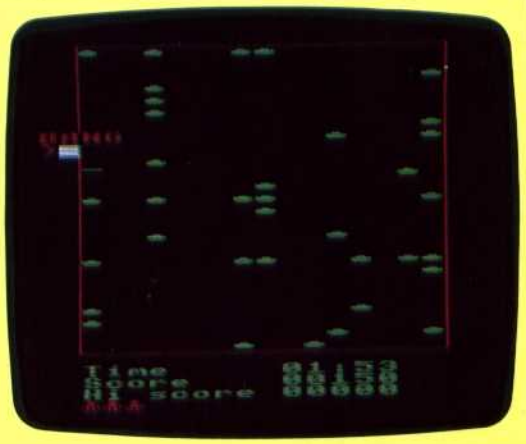
```
3550 DEFPROCexplode(x-explode,y-explode)
3560 PRINT"BANG"
3570 END
3580 ENDPROC
```

Let's finally look at the procedure 'found-mine', called when the detector moves into a cell occupied by a mine. A sound effect to indicate finding a mine would be a nice idea. We shall be looking in more detail at sound later in the project, so for now all we need to know is that the SOUND statement in line 3790 produces a high-pitched 'ping'. The main function of this routine, however, is to increment the player's score. The program uses two variables for the score, the first of which is a numeric variable that is incremented by 150. So that the score is always PRINTed as a five-digit number, we must add leading zeros to its numeric values. In order to do this, we must first convert the numeric value of the score to a string variable and then use string-handling techniques, as described earlier in the project (see page 404) to add on leading zeros. The complete procedure is:

```
3770DEF PROCfound_mine
3780REM ** SOUND EFFECT **
3790SOUND 2,-15,170,3
3800REM ** INCREMENT SCORE **
3810COLOUR 2
3820score=score+150
3830score$=STR$(score)
3840 score$=LEFT$(zero$,5-LEN(score$))+score$
3850PRINTTAB(11,28);score$
3860ENDPROC
```

In the last part of the project, we wrote a short calling program for the procedures we have written so far. The procedures we have given here can be added to your program with the line numbers shown. The only alteration we need to make, at this stage, to get the program to run is to call the procedure 'test-keyboard' from within the main loop of the calling program. Therefore, you will need to add the following temporary line to your program:

```
55 PROCtest-keyboard
```



IAN McKINNELL/LIZ HEANEY

"BANG" On Target

At this point in the Minefield project, the program will fill the screen with randomly placed mines; create your character and a mirror image; define score tables, and provide movement. Because the program is not complete, if you run into a mine you will only see the word "BANG" printed on the screen. In the next instalment of this project, we will design a routine that creates an actual explosion with attendant sound effects. It is also possible that you may encounter error messages at particular points in the game's execution. These messages arise because of the incomplete structure of the game, and will be cleared up as the final subroutines are added in the next instalments. Still, we have reached a point where the program has assumed the characteristics of a fast-action game